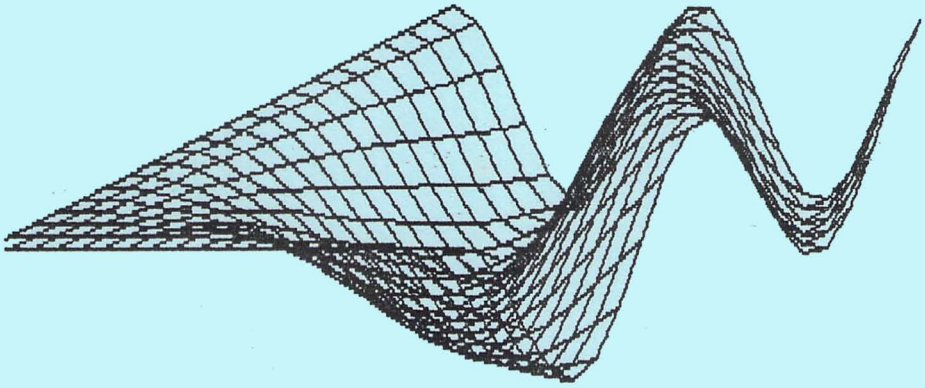
$19.95

ST Math & Graphics

By Brian Hogan



INCLUDES

GFA 3.0 Program Listings

Disk Containing Books Programs

Bonus GFA Subroutine that allows Interactive Input of functions in GFA Programs

FOREWARD

The purpose of this book is two-fold. First, as a mathematician, I'd like to teach some relevant mathematics as it relates to the modern world of computer graphics. Second, as a lover of these "new" high resolution microcomputers, I'd like to present some basic principles of computer graphics. I want to be able to provide the mathematical background as we go along so that you may continue your education in other books more sophisticated than this one.

I'm going to assume at least some minimal background in algebra. Trigonometry would be nice. However, I will give enough of the basic Trig background to allow you to survive the explanations. However, if you really are serious about computer graphics, you should get yourself a high-school level trig book at your "local" bookstore. The rest of the mathematics you will learn traditionally comes from a variety of courses, even Calculus and Linear Algebra! (DON'T PANIC-- we're not going to need Calculus-- not everything in a calculus course is calculus!)

PROGRAM NOTES

The programs in the book are written in GFA BASIC. I did not make full use of the power of this version of BASIC, as my purpose was to demonstrate graphics principles, not the language. For the most part, medium resolution is what I've assumed. In most cases, low resolution would be OK also. Don't be afraid to experiment! I've virtually done nothing with color. Again, the purpose of the book was not to create paint programs or do arcade(game) graphics. Feel free to add some color if you desire to "jazz" up the programs. I didn't want to distract from the purpose of this book and make "hard" looking programs by putting in fancy stuff. Don't

misunderstand me, I love color--I wouldn't dream of not having a color computer--I've owned three of them, from a Compucolor, an Atari 800, and now my present 1040 ST. Most of all my programs that I do for fun involve plenty of graphics and color!

I've included almost all the programs in the book on a disk. You don't know how much I debated over that! I feel there is a great value in typing in programs. I felt I've actually got to basically understand many programs during the process of copying the programs from a magazine article into my computer. The temptation is not to closely look at the programs when they're all done for you. I hope you will attempt to write your own programs and to modify the book's programs--the only way to learn is to be an active participant--programming is not a spectator sport.

MISCELLANEOUS

I've included answers to almost all the problems in the appendix. I've also included in the appendix a couple of other program listings related to the book. (They're on the disk also)

Have fun.

Brian Hogan

Instructor at Highline Community College

Member of S*P*A*C*E computer club

chapter one

POINTS & LINES

Let's start out by defining a few terms. These won't quite be as precisly stated as some of the mathematical definitions that will come later on(but I'm sure you'll forgive me).

PIXEL: The smallest point on the screen that can be illuminated.

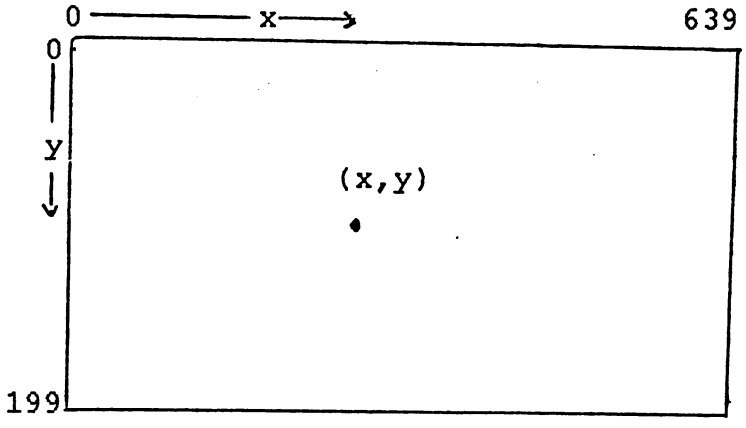
RESOLUTION: The number of pixels that can be put on a screen. This is usually given by stating the number of pixels that will fit across the screen horizontally then giving the number of pixels that vertically go down the screen.

The standard resolution for the ATARI ST depends, as I'm sure you know, on the resolution you choose from the desktop. Low resolution is 320 by 200, medium is 640 by 400, and high resolution(monochrome monitor) is 640 by 400. Other machines may have slightly higher or lower resolutions or they may have several different graphics modes with varying degrees of resolution. (Usually resolution is sacrificed for increased number of colors.) Most of the programs in this book will assume you're in Medium resolution(640 by 200) on the ATARI ST.

PLOTTING A POINT

To plot a point, there must be a coordinate system established for the screen. The usual method on the more popular machines is to establish the "origin" (0,0) at the upper left corner. (To the dismay of the mathematicians). This is also dependent on the language you might be using. GFA BASIC will assume the origin in the upper left corner. TRUE BASIC's location of (0,0) is completely in the hands of the programmer.

A typical screen might be thought of as in the following diagram:(if using GFA BASIC)



To simply plot a single point is not a very illuminating task.(pun intended!!) However, in the interest of learning to walk before running a marathon, let me describe a very simple program. Don't feel you have to type it in however--as I said, its not too exciting.

```
'   Interactive input to plotting points
'            GFA BASIC  VERSION 3
DO
  INPUT "Enter coordinates of point to plot";x,y
  PLOT x,y
LOOP
'   Simultaneously press  control,shift, and alternate keys
'          to exit this infinite loop
END
```

I, on purpose, didn't include any safeguards on your input. What happens if you enter numbers that are outside of the range of the screen?

For example, try inputting 400,5 or -2,7 or .....

You will find that the ST's GEM operating system is sophisticated enough not give you any errors if you try to plot points off the screen. This might not be the case if you tried the same thing on other machines. Well, as you can see just simply plotting a group of points seems rather "pointless"(I'm sorry). Putting

these points in some kind of an order makes things happen. For example:

```
'   Some order in plotting points
'           GFA BASIC
FOR i=10 TO 100
  PLOT i,50
NEXT i
INPUT "",a$  ! press Return to end program
END
```

As you can probably see, this will give us a line from (10,50) to (100,50). By changing the PLOT line, you can get different effects. Try changing the PLOT i,50 to PLOT i,i . Now you will get a diagonal line from (10,10) to (100,100).

problems

In the program above, change only the PLOT line to do the following:(use only PLOT statements that plot a single point.)

1. Plot a vertical line
2. Plot a horizontal line that "moves" from (100,50) to (10,50).
3. Plot a square such that one corner is at (10,10), and the corner diagonally across is at (100,100).(you'll have to "cheat" and replace the single PLOT statement with four PLOT statements)
4. Try these in place of the PLOT i,50 line.:

```
(a)  PLOT i,9*RND(1)
(b)  PLOT i,9*SQR(i)
(c)  PLOT i,SQR(2025-(i-55)*(i-55))
(d)  PLOT i,ABS(55-i)
(e)  PLOT i,45-ABS(55-i)
```

PLOTTING A LINE

If you tried some of the suggested lines in problem 4 above, you will now realize that plotting points can give you some interesting patterns. Of course in such a simple program you are limited to what you can do. Now, if you're just interested in graphing lines, then using the built-in command LINE (or DRAW TO) is faster and more economical programming-wise. Again, let's look

at a simple program:

```
'   Line drawing command in Basic
'           GFA BASIC
DO
  INPUT "Enter endpoints of line(ie. 20,30,40,90)..";x1,y1,x2,y2
  LINE x1,y1,x2,y2  ! DRAW x1,y1 TO x2,y2  performs same action
LOOP
'   Simultaneously press Control,Shift,Alternate keys to escape
END
```

The ST is sophisticated enough that if you give endpoints such that if part of the line is off screen, it will not result in a fatal error. This would not be true of all machines and/or languages.

The program above is not inherently very interesting because there is no built-in "reason" for plotting the lines. Putting the lines on the screen in somewhat of a pattern is what becomes interesting.

Examine the program that follows for an example of plotting lines. Its just a "for fun" program.

```
'   A 'for-fun' program--nested squares
'           GFA BASIC
LET t=1
LET a=0
LET b=45
LET s=3
DO
  FOR i=a TO b STEP s
    COLOR t
    PLOT 10+i,10+i
    DRAW TO 100-i,10+i TO 100-i,100-i TO 10+i,100-i TO 10+i,10+i
    '    BOX 10+i,10+i,100-i,100-i could replace 2 lines above
    '   The coord. are diagonally opposite corners of rectangle
    PAUSE 5  ! Causes the display to slow up--GFA is fast!
  NEXT i
  LET t=1-t
  LET a=45-a
  LET b=45-b
  LET s=-s
LOOP  ! Be sure you track this by hand to see what is happening.
'   Simultaneously press Control,Shift,Alternate keys to escape
END
```

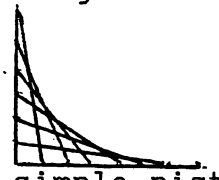
If you run this program you will notice nested squares being drawn and erased, over and over. The erasing is accomplished by

changing the drawing color to the background color. The variable t "toggles" between 1 and 0 . This allows us to change the color each time we repeat the loop. Also the values of a,b,and s are toggled between two values.

There are almost countless "cute" little patterns that involve line drawing. I won't go into more at this time. If you want to try your hand at some patterns try some of the problems below.

problems

1. Change the program above so that there are nested triangles.
2. Make a string art design similar to the following picture.



3. Draw a simple picture, such as a house with windows and a door.
4. Draw a small square and make it move across the screen.

Expanding on these simple programs for a more serious purpose will be the aim of future chapters.

chapter two

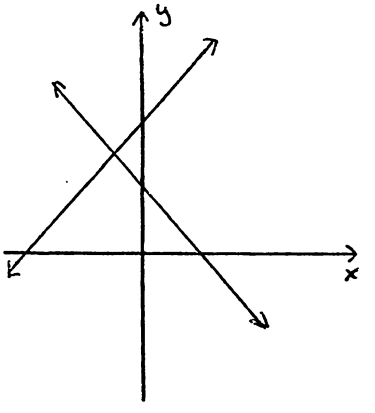
LINES & VECTORS

In this chapter, I'll get down to some serious mathematics. We'll examine a couple of ways to draw lines, and in the process develop some math. The math may not be new to you, it will depend on your background.

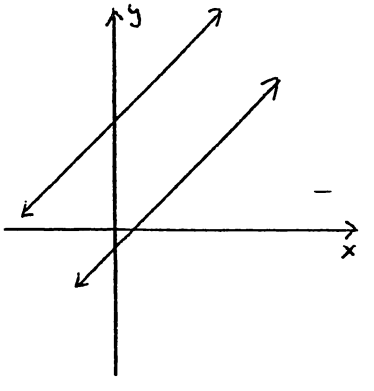
It may be you're asking yourself, "What do I need of knowing how to draw lines--I have a convenient LINE(or DRAW TO) command.?". Well, that may be so in BASIC, but what if you were put in a more primitive environment (such as machine language)? Then it might be handy to know how to start creating your own routine to draw lines. Also, the mathematics we develop will be useful to us to explain other aspects of graphics.

SLOPE OF A LINE

If you draw a line on a piece of paper, you might ask "What characteristics of this line distinguish it from some other line that I might draw?". In looking at other lines you might conclude that certainly the inclination of the line makes it distinguishable. A "flat" line looks a lot different from a line at some angle.

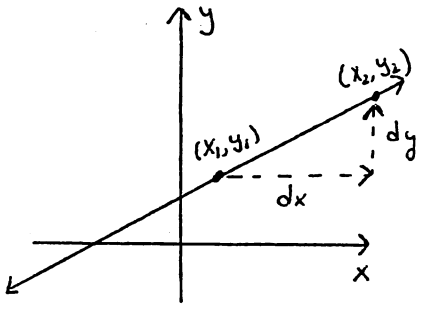


Then if you drew two lines at the same angle, you then would have to conclude that the position of the line makes a difference. To describe the inclination of a line make the following definition:



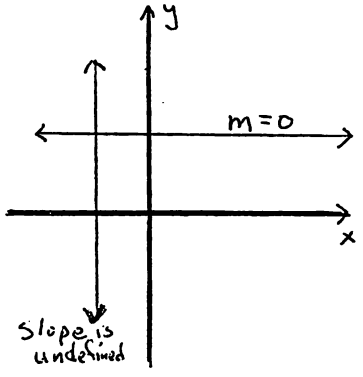
SLOPE OF A LINE = m = dy/dx where

$dy=y_2-y_1$ and $dx=x_2-x_1$



(x_1,y_1) and (x_2,y_2) are any two points on the line. dy is often refered to as the "change in y" and dx is the "change in x". The slope is simply a ratio of the vertical distance to the horizontal distance.

A FEW SLOPES COMPUTED



If the line is horizontal, then the dy is 0 in value, hence the slope m = 0/dx = 0. If line goes at a strict 45 degree angle, then the values of dy and dx would be the same, hence the slope would be 1.

If the line is vertical then the slope is undefined, since the value of dx = 0.(Recall division by 0 is undefined)

Examples:

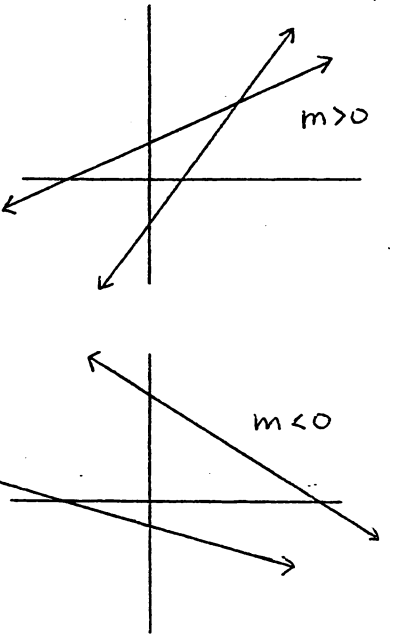
(a) Suppose a line passes through (3,5) and (7,24). Then to find the slope of that line, first calculate dy and dx. dy=24-5=19 dx=7-3=4. Then the slope m=dy/dx=19/4

Note that it would not matter which point we thought of as being (x1,y1) and (x2,y2). If we subtracted in the other order, we get dy=5-24=-19 and dx=3-7=-4. But m=-19/-4=19/4 still!

(b) Suppose a line passes through (3,12) and (7,5). dy=5-12=-7 and dx=7-3=4. Hence m=-7/4. In this case the slope is negative.

If you would take the time to examine the difference in the two examples, you will note that in (a) the line was

increasing upward, whereas in (b) the line was decreasing downward. This will produce different "kinds" of slopes. The "uphill" line will always have a positive slope and the "downhill" line will always have a negative slope. What may be slightly confusing later in these terms "uphill" and "downhill" is when we get to the computer. The y-axis of the computer is upside down(from top left corner down) from the normal mathematics way of thinking of the x and y-axis.



problems

1. Suppose a line L is passing through the following pair of points. Find the slope of line L.

(a) (2,5) and (6,12) (b) (25,14) and (44,2)

(c) (-3,12) and (-5,-10) (d) (5,6) and (12,6)

(e) (-5,12) and (-5,-3)

2. If one point on a line L is (5,8) and the slope is 2/5, give the coordinates of 3 other points on L.

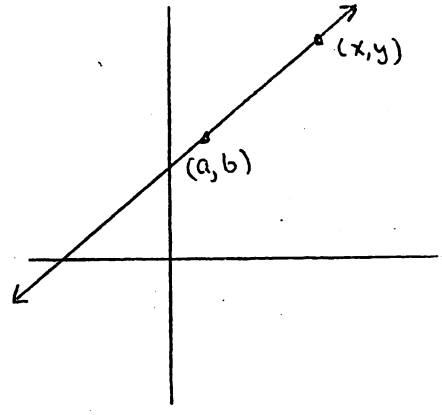
3. If one point on a line L is (-4,5) and the slope is -5/3, give the coordinates of 3 other points on L.

4. If two lines were parallel to each other, what would you conclude was true about their slopes?

EQUATION OF A LINE

In this section will be developed the traditional equation of a line that you may have learned in conjunction with your high school algebra.

Suppose we know a point (a,b) is on a line, and that its slope is m. Let (x,y) be <u>any other</u> point on that same line. Since a straight line could not change its slope any place along the line, it would have to be true that the slope between (a,b) and (x,y) would be m. In other words:



$$\frac{y - b}{x - a} = m$$

or $y-b = m(x-a)$ [point-slope form]

<u>Any</u> point on the line we were talking about would have to satisfy that equation.(ie. if we "plugged" in the x and y coordinates into the equation, we would get a true statement)

Examples:

(a) Suppose a line has a slope of 2 and passes through (3,4). Its equation would be $y-4=2(x-3)$. The point (4,6) is also on the line since $6-4=2(4-3)$ is true. Also (0,-2) is on the line since $-2-4=2(0-3)$ is true. If you only knew one coordinate of a point on the line, you can put it into the equation and <u>solve</u> for the other coordinate.

(b) Suppose a line passes through (-3,5) and has a slope of -3/4. It's equation would be $y-5=-3/4(x+3)$. Suppose you knew the x coordinate of a point on the line was 3. To find the y coordinate, simply solve $y-5=-3/4(3+3)$ for y. You will get $y=1/2$.

ANOTHER EQUATION OF A LINE

Another useful form of an equation is gotten by considering the known point to be the y-intercept. Suppose the line has a slope of m and a y-intercept of (0,b). Then the equation of the line is

y-b=m(x-0) which simplfies to

$$y = mx + b \text{ [slope-intercept form]}$$

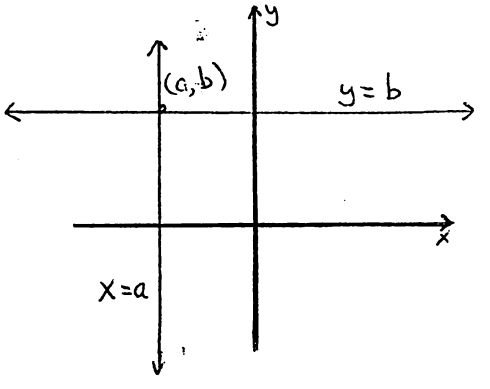
The slope-intercept form is often handy to get information about a line when all we have is an equation in some non-standard form. For example, 3x+2y=9 is an equation of a line. If we solve it for y, we get y=-3/2x + 9/2. In this form we can see that the slope is -3/2 and the line crosses the y-axis at (0,9/2).

By the way, the equation Ax+By=C is sometimes called a "General Form" of an equation of a line.

TWO SPECIAL CASES:

(a) The equation of a horizontal line through (a,b) is y=b.

(b) The equation of a vertical line through (a,b) is x=a.



problems

1. Find the equations of the lines through the given point and with the given slope: (a) point is (3,6); m=-3 (b) point is (-2,5); m=2/3 (c) point is (-2,-4); m=-2/5

2. Find the equation of the following lines: (a) y-intercept of (0,4); slope of -3 (b) horizontal line going through (-4,5) (c) vertical line going through (-3,-5)

3. Determine the slopes from the following equations of lines: (a) 3x+y=4 (b) 2x-3y=9 (c) y=3 (d) y-5=3(x-4) (e) x=4 (f) -2y-3x-10=0

4. Find the equation of the line that goes through the two points: (a) (3,4) and (-5,9) (b) (-2,5) and (2,6) (c) (-4,-2) and (3,-10)

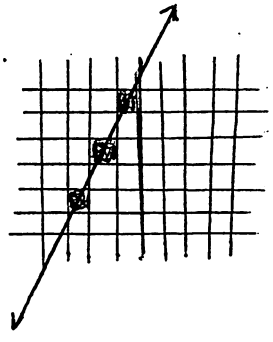
5. Find the equation of a line that passes through (3,4) and is parallel to a line given by the equation 3x+2y=5.

6. A line that is perpendicular to another line has a slope that is the negative reciprocal of the other slope. Find the equation of a line that passes through (-4,5) and is perpendicular to the line 3x-5y=2.

GRAPHING A LINE BETWEEN TWO POINTS ON THE COMPUTER

Now let's use our new-found knowledge to plot a line on a computer screen. The program essentially divides itself into two major parts: graphing a line whose slope is less than one, and graphing a line whose slope is greater than one. To draw a line, we will use the point-slope form of a line, solving it for y: y=m(x-a)+b. The slope will be calculated from the two given points (x_1,y_1) and (x_2,y_2). Then the program will use (x_1,y_1) as the given point (a,b). At first thought, it should just be a matter of using a FOR-NEXT loop from x1 to x2, calculate the y values, then PLOT x,y.

If we don't have a steep line this works fine, but once we get above a slope of 1, moving over 1 in the x direction, may give a jump of more than 1 in the y direction, hence leaving "holes" in the line.(See the figure at the side) So to handle that case, if the slope is steep, then we just solve the equation for x: x=1/m(y-b)+a. Then use a FOR-NEXT loop



from y1 to y2, calculate the x values, then PLOT x,y.

Since plotting a horizontal line or a vertical line doesn't require the use of fancy formulas, they are separate routines. Finally, we include the special case of plotting a single point if both the given points are the same. It seems like a long program, but only a part of it is running at any particular time, hence it executes fairly rapidly.(As fast as you can expect BASIC to run!)

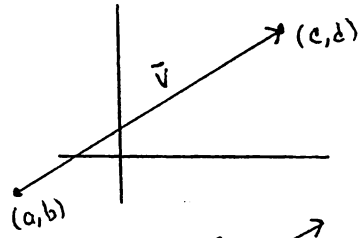
```
' *** plot a line ***
' *** LINEPLOT.V1 ***
' ***  GFA BASIC VERSION 3  ***
'   variables used
'   x1,y1,x2,y2   end points of a line
'   dy,dx  change in y and x coord.
'   m   slope of line
'   rm  reciprocal of slope
'   x,y   points on line to be plotted
'
DO
  INPUT "Input start and end coord.(x1,y1,x2,y2) ";x1,y1,x2,y2
  LET dy=y2-y1
  LET dx=x2-x1
  IF dx=0 AND dy=0 THEN  ! a single point
    PLOT x1,y1
  ELSE IF dx=0   ! a vertical line
    FOR y=y1 TO y2 STEP SGN(dy)
      PLOT x1,y
    NEXT y
  ELSE IF dy=0   ! a horizontal line
    FOR x=x1 TO x2 STEP SGN(dx)
      PLOT x,y1
    NEXT x
  ELSE  ! an oblique line
    LET m=dy/dx
    LET rm=dx/dy
    IF ABS(m)<1 THEN ! not a steep slope
      FOR x=x1 TO x2 STEP SGN(dx)
        LET y=m*(x-x1)+y1
        PLOT x,y
      NEXT x
    ELSE  !steep slope
      FOR y=y1 TO y2 STEP SGN(dy)
        x=rm*(y-y1)+x1
        PLOT x,y
      NEXT y
    ENDIF
  ENDIF
LOOP
' Simultaneously press Control,Shift,Alternate to escape
END
```

GEOMETRIC VECTORS

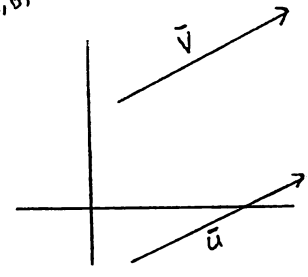
You may well ask, "how does this fit in with what we've just done?". The answer to that question will be to develop some new mathematics and <u>from that</u> develop a different algorithm for graphing a line. So be patient as the new tools are defined and developed. The general topic of vectors will give rise to many other applications in this course, as you will see.

In this chapter, we will be talking only about "vectors" in the plane. The ideas will easily generalize to higher dimensional spaces.

DEFINITION: A vector $\bar{v}$ from points (a,b) to (c,d) is denoted by $\bar{v} = \langle c-a, d-b \rangle$. The values of c-a and d-b are called the components of $\bar{v}$.



DEFINITION: Two vectors are equal if and only if their respective components are equal.

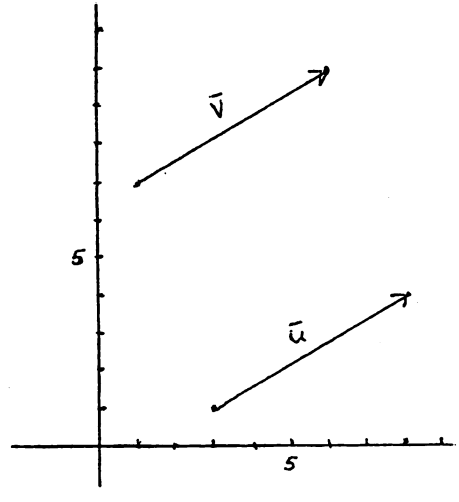


Examples:

(a) The vector $\bar{v}$ from (1,7) to (6,10) is $\langle 6-1, 10-7 \rangle = \langle 5,3 \rangle$

(b) The vector $\bar{u}$ from (3,1) to (8,4) is $\langle 8-3, 4-1 \rangle = \langle 5,3 \rangle$

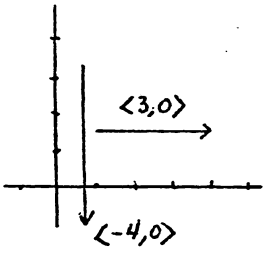
(c) The vectors in examples (a) and (b) are equal since their components are equal.(ie. $\bar{u} = \bar{v}$)



(d) The vector from (8,4) to (3,1) is $\langle 3-8, 1-4 \rangle = \langle -5,-3 \rangle$. This is a different vector than in example (b).

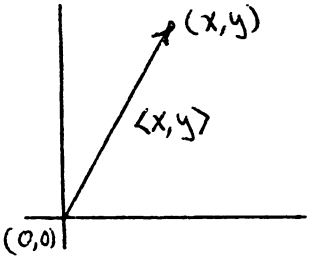
Geometrically one represents these geometric vectors as "arrows"

from the first point to the second point. By drawing a few pictures it soon becomes evident that equal vectors are arrows that are parallel to each other and have the same length. If a vector goes in a different direction or is a different size, then the vectors will not have the same components.



Be sure to examine the "picture" of a vector that has a component of 0. These horizontal and vertical vectors can be thought of as the building blocks of <u>any</u> vector as you will see later.

In the previous example, $\bar{u} = \bar{v} = \langle 5,3 \rangle$. Note that a vector from the origin (0,0) to the point (5,3) would be equal to those two vectors also. Thus a point (x,y) can be thought of as representing a vector <x,y> that is drawn from the origin to the point. Because of this one to one correspondence between points and vectors from the origin, you find many books use the same notation for a point and a vector.



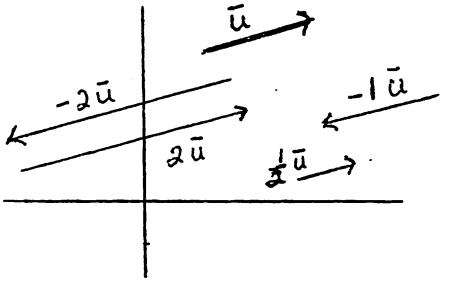
I think this is unnecessarily confusing at the beginning, so I will use the notations as established in this book.

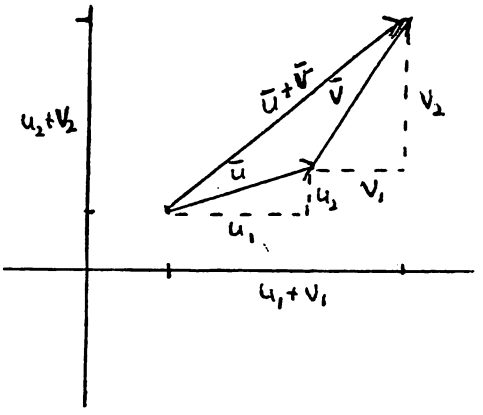
Example: Suppose a vector "begins" at (3,4) and has components of <-2,4>. What is the coordinates of the "ending" point?

Solution: Let (x,y) be the unknown ending point. Thus <-2,4> = <x-3,y-4>. Therefore -2=x-3 and 4=y-4 (Equal vectors have equal components). Solving these, we get x=1 and y=8. So the ending point is (1,8).##

DEFINITION: Let $\bar{u}=\langle u_1,u_2 \rangle$ be a vector and let t be any real number(called a scalar) then

$t\bar{u}$ is the vector $\langle tu_1,tu_2 \rangle$.

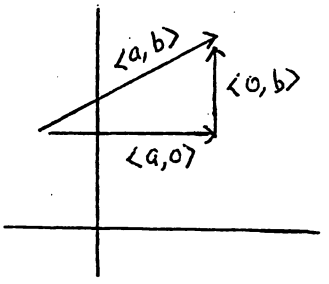




DEFINITION: If $\bar{u}=\langle u_1,u_2\rangle$ and $\bar{v}=\langle v_1,v_2\rangle$ are vectors, then $\bar{u}+\bar{v}$ is the vector $\langle u_1+v_1,u_2+v_2\rangle$.

Note: $\langle a,b\rangle = \langle a,0\rangle + \langle 0,b\rangle$ by this definition.

Examine the picture to the right to see the geometric interpretation of adding vectors.



As you can see if you connect the vectors up, beginning to end, the vector you draw from the beginning of the first vector to the end of the last vector is the vector $\bar{u}+\bar{v}$.

Note that a single vector is just geometrically adding up the horizontal and vertical vectors made up from its components.

problems

1. Let $\bar{u}=\langle 2,3\rangle$ $\bar{v}=\langle 3,-7\rangle$ $\bar{w}=\langle -3,4\rangle$

   Find (a) $\bar{u}+\bar{v}$ (b) $3\bar{w}$ (c) $\bar{v}+\bar{w}$ (d) $-2\bar{v}$ (e) $2\bar{v}+3\bar{w}$

2. Find the vector from (3,-4) to (-7,-10)

3. A vector $\langle -3,7\rangle$ ends at (4,-3). Find the beginning point.

4. A vector $\langle 4,-3\rangle$ begins at the point (-4,-5). What is the ending point?

5. A vector $\langle 5,2\rangle$ begins at a point (x,y) and ends at the point (2y,x). Find the values of x and y and the two points.

6. <u>Prove</u> that $\bar{u}+\bar{u}=2\bar{u}$, for any vector $\bar{u}$.

7. Sketch $\langle 3,4\rangle$, $2\langle 3,4\rangle$, $-2\langle 3,4\rangle$, $1/2\langle 3,4\rangle$. Assume all vectors start at the same point. If $\bar{u}=\langle 3,4\rangle$, make some general conclusions about $t\bar{u}$.

8. What would the vector $\langle 0,0\rangle$ represent?

9. Suppose $\bar{u}=\langle a,b\rangle$. What formula would represent the length of the vector $\bar{u}$?

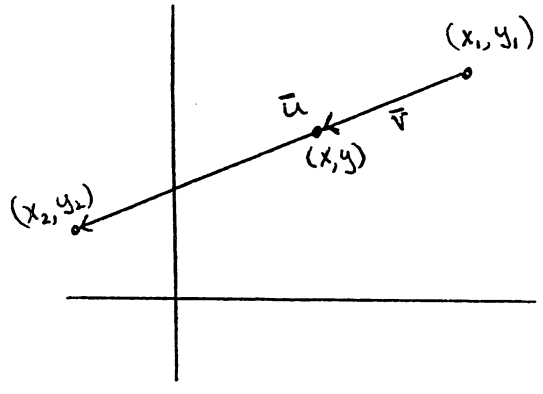
EQUATION OF A LINE(AGAIN)

Although we did not formally prove the fact, the results of the previous section(definitions, discussion and problems) should have left you with the following theorem.

THEOREM: $t\bar{u}$ is a vector that is vector parallel to the vector $\bar{u}$, and the length of $t\bar{u}$ is t * length of $\bar{u}$. If t is positive, $t\bar{u}$ is a vector in the "same" direction as $\bar{u}$ and if t is negative, $t\bar{u}$ is in the opposite orientation.

(You should try to prove this. Make up a definition of the length of a vector. How are you going to get a handle on the "direction" concept?)

OK, let's consider getting an equation of a line between two points (x_1,y_1) and (x_2,y_2).



Let $\bar{u} = \langle x_2-x_1, y_2-y_1\rangle$ be the vector from (x_1,y_1) to (x_2,y_2). Let (x,y) be a point on the line segment between the two points. Let $\bar{v} = \langle x-x_1, y-y_1\rangle$ be the vector from (x_1,y_1) to (x,y).

"Obviously" $\bar{v}$ is a vector parallel to $\bar{u}$, in the same direction and shorter than $\bar{u}$. Thus $\bar{v}=t\bar{u}$, $0\le t\le 1$.

So,

$$\bar{v} = t\bar{u}$$

$$\langle x-x_1, y-y_1 \rangle = t\langle x_2-x_1, y_2-y_1 \rangle$$

$$\langle x-x_1, y-y_1 \rangle = \langle tx_2-tx_1, ty_2-ty_1 \rangle$$

Hence,

$x-x_1 = tx_2-tx_1$ and $y-y_1 = ty_2-ty_1$

$$\boxed{x = x_1+(x_2-x_1)t \quad \text{and} \quad y = y_1+(y_2-y_1)t}$$

The above pair of equations, $0 \le t \le 1$, are called the parametric equations of a line from (x_1,y_1) to (x_2,y_2). t is called the parameter.

Example: Find the parametric equations for a line from (3,5) to (8,15).

Solution: $x=3+(8-3)t$ and $y=5+(15-5)t$, $0 \le t \le 1$. Simplifying we get $x=3+5t$ and $y=5+10t$, for $0 \le t \le 1$. ##

problems

1. What are the parametric equations of the line segment from (2,-3) to (7,4)?

2. What are the parametric equations of the line segment from (-3,-7) to (-50,-8)?

3. What are the parametric equations of a vertical line or a horizontal line.

4. For the parametric equations $x=x_1+(x_2-x_1)t$ and $y=y_1+(y_2-y_1)t$, $0 \le t \le 1$, what point do you get if t=0? if t=1?

5. For the line segment given by $x=7+8t$ and $y=4-3t$, $0 \le t \le 1$, what were the two endpoints?

6. For the same equations in prob. 5, find the intersection point of the line segment with the horizontal line y= 2. Do the same for y=-2.

7. If we allow t to take on other values besides those between 0 and 1, what points do we get? Generalize your conclusions in a statement.

8. Use parametric equations to prove the midpoint between (x_1,y_1) and (x_2,y_2) is $((x_1+x_2)/2, (y_1+y_2)/2)$.

GRAPHING A PARAMETRIC EQUATION OF A LINE

I hope you recognize the relative ease in programming a computer to graph a line point by point by these set of parametric equations. Basically a FOR-NEXT loop using the variable t, t going from 0 to 1. Obviously we need to figure out the appropriate STEP size so as to get a solid line. To see what this step value is, see what value of t is needed to increase the value of x and y no more than plus or minus 1 from the starting point (x_1,y_1) (ie. t=0 for this point).

I'm sure you don't want me to spoil your fun of writing this short program. Stop reading this book and write a <u>short</u> program that allows you to INPUT the endpoints of the line segment(within the boundaries of your screen coordinates), and the program will graph a line between the two points.

chapter three
WINDOWS AND VIEWPORTS

In this short chapter, we will develop one of the main mathematical relationships between the "real world" and the computer screen. One main purpose of computer graphics is to represent real and mathematical objects on a computer. The following definitions are usually used in this topic:

DEFINITION: A closed interval [a,b] = { $x \in R$: $a \le x \le b$ }

DEFINITION: A cartesian product A x B.

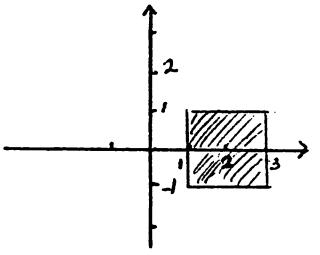
A x B = { (a,b): $a \in A$ and $b \in B$ }

DEFINITION: [m;n] or ([m:n]) = { $x \in$ Integers: $m \le x \le n$ }

Example:

$[1,3]x[-1,1] = \{ (x,y): x \in [1,3]$ and $y \in [-1,1] \}$

(See the picture to the right.)



Example: [0;639]x[0;199] represents the ST Medium Res screen.

[0;319]x[0;199] represents the ST Low Res screen.

The above are sometimes called Screen Domains.

Example: RxR (often shortened to R^2) represents the plane. (Often referred to as the Cartesian Plane.)

problems

1. On a piece of graph paper sketch the graph of the following sets:

(a) [-1,3]x[-2,2] (b) [-3,-1]x[-2,0] (c) [-2;3]x[1;4]

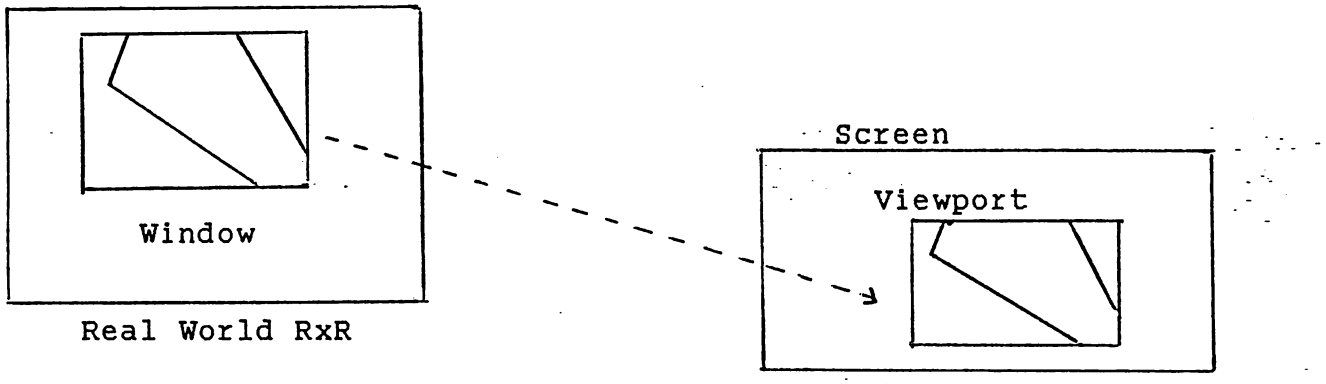
2. What would be the screen domain of the monochrome ST High resoulution mode?

3. What would you guess the definition of AxBxC was?

DEFINITION: A window is a rectangular subset of RxR.

DEFINITION: A viewport is a rectangular subset of the Screen Domain.

It is important you have the following diagram in your mind to keep straight these two definitions. They are often confused!! BEWARE-- even some texts are at times careless in the use of these two words. The confusion in terms comes from the fact that the viewport is the computer's representation of the window. *Hence* the little rectangles on the computer screen are sometimes also called windows by computer companies(Including Atari, Microsoft, etc). However, for the mathematics of the situation, we need to keep the two concepts separate. Maybe the following diagram below will clarify the situation.

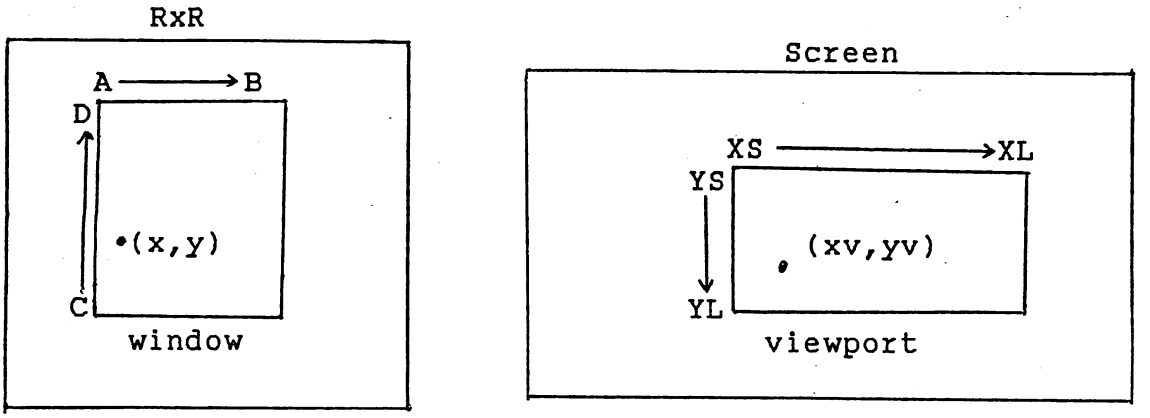


DEFINITION: The function that takes points from the window to the viewport is called a windowing function.

Objects in the viewport may look distorted when compared to how they look in the window. However it is normally assumed any points in the viewport occupy the same relative position as they did in the window. For example, a point in the window that was located 1/3 of the way between the two horizontal sides and 1/2 of the way between the two vertical sides, would occupy the same relative position in the viewport. In other words, "proportionality" is preserved.

With the above in mind, look at the diagrams and the resulting ratios. I'm assuming the "normal" real-world orientation of the smaller y values below the larger values(just the opposite of

most screen coordinate systems which assumes the smaller y values are toward the top of the screen). I'm assuming the window is [A,B]x[C,D] and the viewport is [XS;XL]x[YS;YL]



$$\frac{x-A}{B-A} = \frac{xv-XS}{XL-XS} \qquad \frac{y-C}{D-C} = \frac{YL-yv}{YL-YS}$$

Solving these for xv and yv(the screen coordinates) we get:

$$xv = \frac{x-A}{B-A}(XL-XS)+XS \qquad yv = \frac{y-C}{D-C}(YS-YL)+YL$$

Now as a double check on these formulas let x=A and y=D. This point should be in the upper left corner of our viewport (XS,YS). Note, indeed, that xv=XS and yv=YS.

The formulas simplify quite a bit in certain special cases. For example, suppose the whole screen is the viewport(XS=YS=0, XL,YL are max. screen resolution). Then we get:

$$xv = \frac{x-A}{B-A}XL \qquad yv = \frac{D-y}{D-C}YL$$

Be sure you see how these formulas are derived. Don't try to memorize these--your brain is worth more than that!

problems

1. How would the mapping formulas simplify if you assumed the window was [-F,F]x[-G,G] (F>0 AND G>0) and the viewport is the whole screen.(Note that our window will put the origin in the

middle of the screen.)

2. What would the general mapping formula be if you assumed that the real world had the same "upside-down" orientation that the screen has.(ie, larger y coordinates are below the smaller y coordinates)

A WINDOW-VIEWPORT PROGRAM

Let me present a program that uses the tools we have so far. I'm still going to pretend we don't have any line drawing commands in our language. Thus, I'll have to use our line drawing routines presented in chapter 2 as a subroutine. I won't actually write in the routine, you can substitute in either the routine given in the book, or the one you wrote using the parametric equations of lines.

The program will just be a simple demo. First, the user will be asked for the window, then the user will give the viewport. Lines will be drawn on the screen outlining the viewport. Then the user will enter the endpoints of line segments(in real world window coordinates) and see the line drawn in the correct relative position in the viewport. The program then asks for more lines to be input. To keep the program simple, no provision has been made to end this infinite loop. Certainly feel free to "jazz" this up to your heart's content.

```
' *** Plot a Line in a Window
' ***    WINDOW.V1
' ***    GFA BASIC  VERSION 3
'
' *** variables used--main program
'
'  a,b,c,d    represents window
'  xs, xl, ys, yl represents viewport
'  xw, yw  is point in the window(local variables)
'  xv,yv   is corresponding point in viewport
'  e,f,g,h  endpoints of line in window
'  x1,y1,x2,y2  endpoints of corresponding line in viewport
'  s,t,i,dx,dy   miscellaneous  variables
PRINT AT(1,20);
INPUT "Enter window  A,B,C,D --[A,B]x[C,D] ",a,b,c,d
INPUT "Enter viewport XS,XL,YS,YL--[XS,XL]x[YS,YL] ",xs,xl,ys,yl
'
' Now draw outline of the viewport
linedraw(xs,ys,xs,yl)
linedraw(xs,yl,xl,yl)
linedraw(xl,yl,xl,ys)
linedraw(xl,ys,xs,ys)
'
DO   ! now draw user lines in viewport
  PRINT AT(1,20);
  FOR i=1 TO 5
    PRINT STRING$(79," ")
  NEXT i
  PRINT AT(1,20);
  PRINT "window = [";a;",";b;"]x[";c;",";d;"]"
  INPUT "Enter ends of line (x1,y1);(x2,y2)--x1,y1,x2,y2 ",e,f,g,h
  convert_to_screencoordinates(e,f)
  LET x1=xv
  LET y1=yv
  convert_to_screencoordinates(g,h)
  LET x2=xv
  LET y2=yv
  linedraw(x1,y1,x2,y2)
LOOP
'  Below are the subroutines used
'
PROCEDURE linedraw(x1,y1,x2,y2)  ! these x1,y1,x2,y2 are local
  '  vector method--assumes screen coordinates
  LOCAL t,s,dx,dy
  LET dx=x2-x1
  LET dy=y2-y1
  LET s=1/(MAX(ABS(dx),ABS(dy))+1)
  FOR t=0 TO 1 STEP s
    PLOT x1+(dx)*t,y1+(dy)*t  ! parametric equations of line
  NEXT t
RETURN
'
PROCEDURE convert_to_screencoordinates(xw,yw)
  ' returns values of Xv and Yv, screen coordinates
  LET xv=(xw-a)/(b-a)*(xl-xs)+xs
  LET yv=(yw-c)/(d-c)*(ys-yl)+yl
RETURN
```

OTHER PROGRAMMING SUGGESTIONS: You might do one or more of the following:

(a) Rewrite the input of the previous program so that one only enters , say,A and B, for a window of [-A,A]x[-B,B]. Simplify the corresponding window function.

(b) Rewrite program so the entire screen is the viewport. Simplify the corresponding window function.

(c) Allow entry of any endpoints("real-world" coordinates) of a line, but only the points that would be in the window(or viewport) would be graphed.

chapter four

FUNCTIONS AND GRAPHING FUNCTIONS

FUNCTIONS

For most purposes we can think of a function as a special type of an equation that is written in such a way that there is only one output for each input. Traditionally, when working with numbers, X is often the variable used for the input, Y the output. The slope-intercept form of a line, y=mx+b, is in that form--input a value for x, and out comes a value for the y coordinate. The more general point-slope equation of a line needs to be solved for y, in order to think of it as a function: y-b=m(x-a) is equivalent to y=m(x-a)+b.

Mathematicians often think of functions in a more general way than just described. The only requirement is that a particular "input"(might not be just a number) results in a single "output"(again, it might not be just a number). A couple of examples that we have met in previous discussions might illuminate that remark.

Example: The parametric equations of a line $x=x_1+(x_2-x_1)t$ $y=y_1+(y_2-x_1)t$ can be thought of as function that accepts a single input for t, then outputs a single point (x,y). The input is a number, the output is a point.

Example: The windowing function accepts a point (x,y) in the window as input, and outputs a single point (xv,yv) in the viewport.

If you're willing to carefully define by what you mean as a "single input" and what is meant by your corresponding output, you'd be surprised how many things can be looked upon as functions!

Often times it is convenient to employ special function notation

in describing your function. One nice feature would be to be able to "name" your function. Saying "y=" can be awful undescriptive, especially if you're talking about several functions at one time.

The special function notation goes in the following format:

name(input)=output

Example: f(x)=2x-5 f is the name of this function, a value for x is to be input, the value of the expression 2x-5 will be the corresponding output.

Example: g(t)=(t-5)/(t-4) g is the name of this function, a value of t is to be input, the value of the expression (t-5)/(t-4) is the output

Example: p(t)=(3+5t,2-4t), $0 \le t \le 1$ p is the name of the function that accepts values of t for input and outputs a point on the line segment between the points (3,2) and (8,-2). (Did you recognize the parametric equations of x=3+5t and y=2-4t?)

Now, if you knew the functions that had been defined, and someone mentioned function g, for example, you'd know exactly which function being referred to.

The names f,g,h,etc are often used as kind of temporary names. You would not be expected to know what function f was unless you had seen it recently defined. Other names are more permanent and universal in their usage.

Some common names are: cos(x), sin(x), log(x), ln(x), ... These are names for some trigonometry functions and logarithm functions found on most scientific calculators. Sometimes special function notation is used, such as $|x|$ is a common symbol for the absolute value function. In BASIC we use ABS(x). Another special notation is the square root function, $\sqrt{x}$. In BASIC we use SQR(x).

The function notation has another big advantage: namely in being able to indicate that a particular input is to be actually used in the function.

Example: Suppose I define f(x)=2x+3. Then if I write f(3), this is to indicate the value of the function f, when 3 is used for the input. Thus f(3)=2(3)+3=9. Similarly, f(1)=5; f(-3)=-3; f(-10)=-17; f(a)=2a+3; etc.

Example: Suppose f((x,y))=(5x+y,2x-y). Then if I write f((3,4)), this is to indicate the output when the input is (3,4). In this case, f((3,4))=(5*3+4,2*3-4)=(19,2). Thus the input of the point (3,4) into the function f is the point (19,2).

problems

1. Look in your BASIC manual. What other functions are predefined and are part of the language?
2. What function in BASIC accepts a number and outputs a character?
3. What function in BASIC accepts an input of a "string" and outputs the number of characters in that string?
4. What function in BASIC accepts a pair of numbers for input and outputs a lighted "dot"(pixel) on the monitor screen?
5. Describe how a SUBROUTINE might be viewed as a function?
6. Let f(x)=3/x. Find (a) f(2) (b) f(-3) (c) f(1/3)
7. Let f(t)=t^2+3. Find (a) f(3) (b) f(-3) (c) f(1.2)
8. Let f(v)=(v,v). Find (a) f(2) (b) f(4) (c) f(-3))
9. Let g(c)=(2c,4-c). Find (a) g(3) (b) g(-3) (c) g(p)
10. Let w((x,y))=(3x-2y,xy). Find (a) w(2,1) (b) w(-3,1) (c) w(0.5,0.7)
11. Let f(x)=SQR(x). Find (a) f(3) (b) f(-4) (c) f(16) (d) f(6.7)

12. Let f(x)=SIN(x). Find (a) f(0) (b) f(1.5) 13. Make up a function that gives you the points on a line between (3,4) and (-4,5).

14. Make up a function that takes in a number as input and outputs twice that number increased by 7.

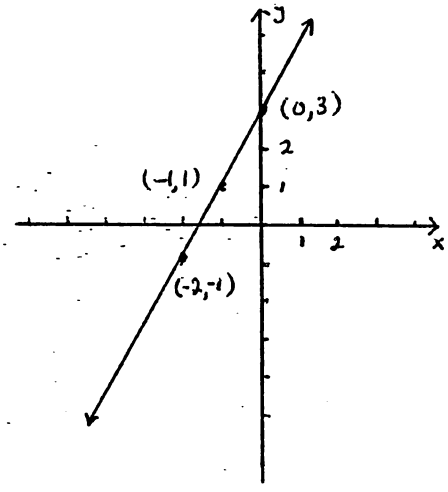
15. Make up a function that takes in a point as input and outputs the distance that point is from the origin.

16. Make up a function that takes in a pair of points as input, and outputs the vector from the first point to the second point.

GRAPHING A FUNCTION

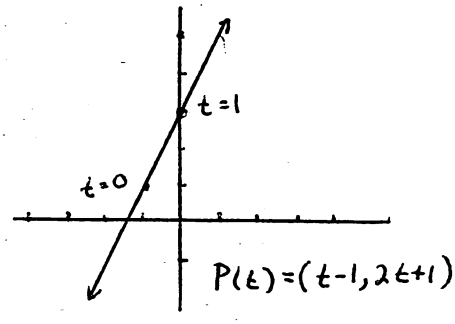
Graphing a function "usually" implies a function that has numerical input and output. The input and output are often displayed as a point (input,output) (WARNING: NOT ALWAYS!!)

In graphing a function that has a single number input and a single number output, often the Cartesian(rectangular) coordinate system is utilized. The input is the x coordinate of the point, the output is the y coordinate of the point. See the example of graphing f(x)=2x+3 to the side.(This is obviously a line, since it is the form of y=mx+b.)



In graphing a function like our parametric equations of a line, often the input is supressed, and all the graph displays is the output(which is a point).

Sometimes additional labels on the graph might indicate a few typical values of the parameter t. See the example to the side.



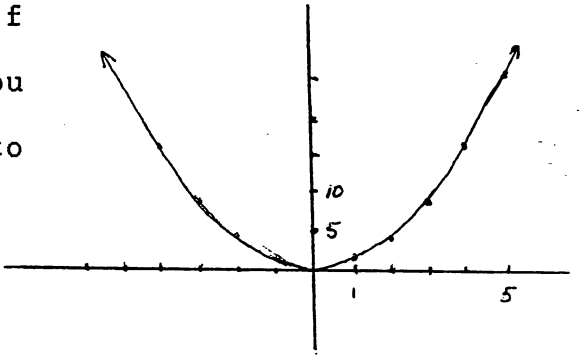
Later in the book, we will discuss different coordinate systems, discuss the concept of parametric functions in more detail, etc.

For now, let's concentrate on functions that are of the form f(number)=number. Drawing graphs of these functions are often rewarding and exciting. We've already looked at the function that gives a line. What about functions like $y=x^2$ ($f(x)=x^2$); y=sin(x) (f(x)=SIN(x)); y= (x-1)/(x +1); or .....

Also, drawing these functions by hand can be very time-consuming and frustrating. It involves either knowing beforehand what the equation will give you as a result, or else computing a lot of points to put on the graph.

Example: Graph the function $y=x^2$. If you don't know what it looks like, you need to compute some points. One way to organize your work is to make a chart:

x	-4	-3	-2	-1	0	1	2	4
y	16	9	4	1	0	1	4	16



The numbers in the chart give you the points (-4,16), (-3,9), ...(4,16). Of course this is only a small insignificant fraction of the possible points. Hopefully it gives the person trying to graph the function a good idea of how the function looks. If the results are still too rough, pick more points; maybe pick some fractional values for x, etc.

Ok, how could the computer help us? One way is to use it to construct our points to be plotted:

```
FOR x=-10 TO 10 STEP .25
  y=x^2
  PRINT "(";x;",";y;")"
NEXT x
```

(In GFA BASIC you could also do the following:)

```
DEFFN square(x)=x^2
FOR x=-10 TO 10 STEP .25
  PRINT "(";x;",";FN square(x)
NEXT x
```

Presto, in an instant you've got more points than you know what to do with!!

But, obviously, this isn't the end of the story(Did you notice the title of the book?). Since we have the commands available in BASIC to plot points and we know how to put those points in any viewport on the screen we wish, why stop with the generation of the points? Let's PLOT those little "buzzards" and have the graphing done for us!!

A GRAPHING PROGRAM

I'll outline a way to do a graphing program. The program I wrote is in the appendix. I don't want to tempt you to take the easy way out and not do the program yourself. It's really quite a short program. If you think you can do one by yourself, quit reading and DO IT!! If not, read on to see how I did it (I lay no claim that it's the best way!). The program I'll describe is not really the one I'd like to do. These newer BASIC's do have some drawbacks compared to some of the older BASIC's with line numbers. For example, I used to be able to say LIST 600, and it would display what was in line 600 of my program. I could even do this while my program was running. Thus if my line 600 was 600 Y=COS(X) it would list that trig function. If that was the function I was graphing (or about to graph) I could display it on the screen while the program was running. I could ask the user if that was the function they wanted to graph. If not, I could stop the program and have them enter in a new line 600 with a different function. Then I could start up the program again, by saying RUN

100(for example), and the program would continue at line 100 . I could work it in such a way that previous graphs were not automatically erased. In other words, I had a kind of an interactive function input in my program. I find it almost impossible to duplicate this in any simple fashion in GFA BASIC. This is unfortunate, in that it is often instructive to compare different graphs overlaid on top of each other. Well, enough crying for what I "ain't got"!! We'll write a program where you must edit the program before you run it in order to get the function you want to graph. The program will allow you to see that function plotted in different viewports with different windows so you can see different parts of the graph on the same screen.

1. At the beginning of my program, I have these statements that contains my function.

```
LET f$="Cos(X)"
DEFFN f(x)=COS(x)
```

This can contain any function. I put the function in a string also, so I can let user know what function is in the program. If you wish to graph another function, stop the program, edit these two lines to contain the desired function, then restart the program.

2. Now, back to the design of the program. It might be nice to reserve the bottom 4 or 5 lines of the screen for a "text" window. We're going to need some place to display the current values of the viewport and window, and then ask for input about new values of the viewport and window variables. You might consider the screen looking, as follows, when the program first starts running.(Of course this all just personal taste--feel free to be creative!!)

3. I'd display the current value of the viewport, ask if that was ok; if not, then ask for new values of the viewport. Do the same for the window. Then for jazziness, draw a box on the screen that represents the window.(Use the built-in line drawing commands.) Just a simple

BOX xs,ys,xl,yl command will do the job. (If you look this command up in your manual, you'll find you need to give the diagonal coordinates of the rectangle.)

4. Next draw in the x-axis and y-axis if they belong in the window(viewport). This is simply done by determining if 0 is an element of [A,B] and [C,D]. If the axis do belong, you'll have to convert the window value of 0 into its appropriate viewport coordinate by use of the windowing function. For example, if 0 is between A and B, then there should be a y-axis showing. If you give the command

DRAW -a/(b-a)*(xl-xs)+xs,ys TO -a/(b-a)*(xl-xs)+xs,yl

The complicated expression is simply what you get when 0 is substituted for x in the windowing function. Now do a similar thing to plot the x-axis(if there is one that should be showing).

5. Now graph the function. This is a simple

```
FOR x = a TO b STEP ??  !Put in appropriate step value
LET y=FN f(x) !get y-coordinate from our equation

'   CHECK TO SEE IF Y VALUE IN WINDOW
'   IF IN WINDOW, CONVERT TO VIEWPORT  Put in appropriate
'   COORD. BY WINDOWING FUNCTION          code!!

PLOT xv,yv
NEXT x
```

That's it! Of course, after this you'll ask if they want more graphs. If things are done right, you should be able to pick a different viewport and graph the function in a different place on the screen. If the user picks fairly small viewports, she could have 4 or 5 different graphs on the same screen. It really looks

quite nice.

Well I hope that gets you going. If you want to compare yours with mine, check my program in the appendix.(Or examine it on the disk) I'm sure yours will be much better! I'll also include on the disk a little more jazzed up version that uses the mouse to define the position of the viewport.

Just a closing remark--it would really be nice if put text on your axis to indicate the beginning and ending values on your x and y axis(ie. window values). The nice thing about the ST is that it is easy to mix text and graphics. See your DEFTEXT and TEXT command in your manual.

I decided not to go into detail about that kind of stuff as it kind of covers up the purpose of the program and this book. By adding all of the "goodies" you could make this program a very powerful tool.(Some of the goodies would be interactive function input, labeling the axis, color, perhaps a little sound, a screen dump to a printer, etc.

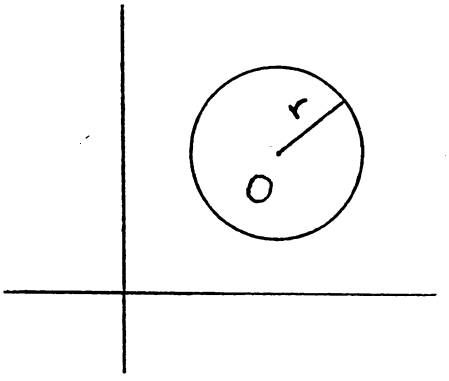
chapter five

ESSENTIAL TRIGONOMETRY

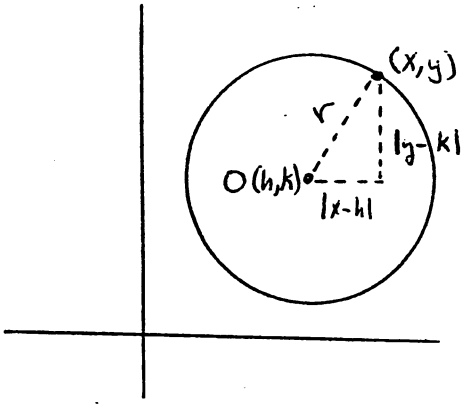
I fervently hope you've had some exposure to trigonometry. It's going to make this a lot easier. If you don't, I hope what I have to say in this chapter will get you to your nearest bookstore to buy a trig. book.(Or enroll in a course at your local "college") I'll try to present the most relevant parts of trigonometry as it will relate to the rest of this book. I only hope I can convey the importance of these functions in such a short space.

EQUATION OF A CIRCLE:

I'm sure you will agree that a reasonable definition of a circle of radius "r" and a center at some point O, is that it is a set of points in the plane that are all at a distance r from the point O.(See picture to the side)



Let's give the point O coordinates (h,k) and pick some arbitrary point on the circle and call the coordinates (x,y). By what I said above, the <u>distance</u> between these two points is r. Using the Pythagorean theorem you get the following equation:

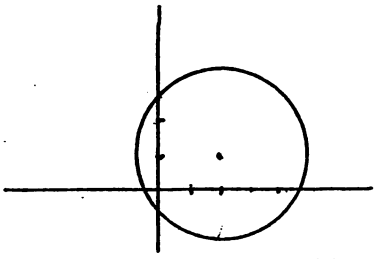


$$(x-h)^2 + (y-k)^2 = r^2$$

(the difference in the x coordinates give the horizontal distance between the two points, similarly the difference in the y coordinates give the vertical distance)

Example: The equation of a circle of radius 5, centered at (3,-4), is $(x-3)^2 + (y-(-4))^2 = 5^2$ or $(x-3)^2 + (y+4)^2 = 25$

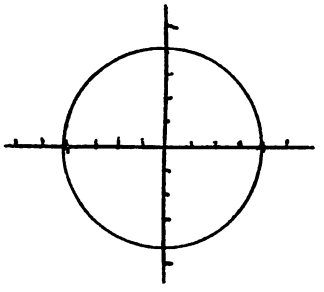
Example: If the equation $(x-2)^2 + (y-1)^2 = 9$ is given, you know it is a circle centered at (2,1) with a radius of 3.



A special form of this equation is when the center is (0,0). Then the equation is simply

$$x^2 + y^2 = r^2$$

Example: $x^2 + y^2 = 16$ is a circle centered at the origin with a radius of 4.



problems

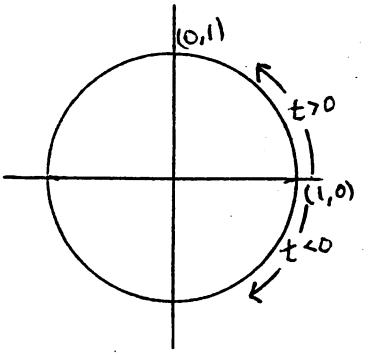
1. Give the equation of the following circles:
   (a) center at (5,5); radius is 10
   (b) center at (-3,6); radius is 2
   (c) center at (-4,-6); radius is 1/2
   (d) center at origin; radius of 1
   (e) center at origin; radius of 5
2. Give the center and radius of the following circles:
   (a) $x^2 + (y-2)^2 = 36$
   (b) $(x-4)^2 + (y+3)^2 = 17$
   (c) $x^2 + y^2 = 8$
   (d) $(x+3)^2 + (y-4)^2 = 64$
3. If (3,5) and (7,9) are endpoints of the diameter of a circle, what is its equation?
4. If $x^2 + y^2 = 4$ is the equation of a circle, and if the x-coordinate of a point on the circle is 1, what is the y-coordinate?; if y-coordinate is 1.5, what is the x-coordinate?
5. If $(x-2)^2 + (y+3)^2 = 10$ is a circle, and the x-coordinate of a point on the circle is 2, what is the y-coordinate?

TRIGONOMETRIC FUNCTIONS

There are two main ways to define the trigonometric functions. I'll present both for comparison, as each has their own special advantages and application. The two methods are totally equivalent though.

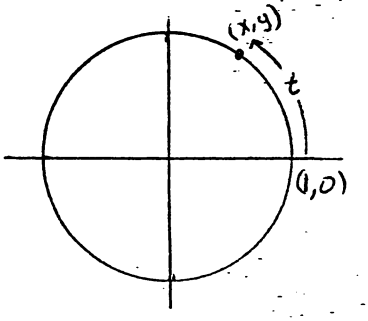
The Unit Circle Definition:

Let $x^2 + y^2 = 1$ be the equation of a circle at the origin, with radius of 1.(Often called the unit circle). Let t be a number representing distance traveled along the unit circle from the point (1,0).(see diagram to the side) If t>0, move counter-clockwise; if t<0, move clockwise. The point on the unit circle you arrive at has some coordinates (x,y).

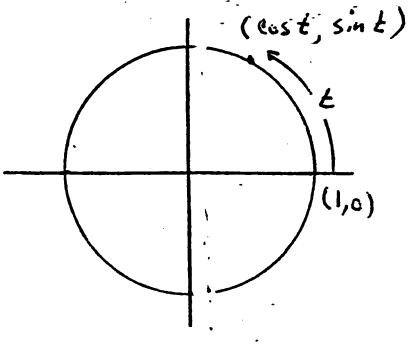


DEFINITION: $\cos(t) = x$

$\sin(t) = y$

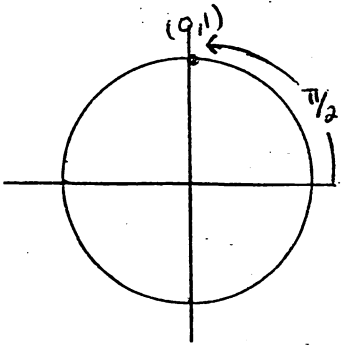


These are called the cosine and sine functions respectively.



Example: If t=0 then the point is (1,0). Thus the cos(0)=1 and sin(0)=0. If you've got a calculator, check it out--Be sure you're not in DEGree, but in RADian mode. Or use your computer. Type in

? COS(0),SIN(0)

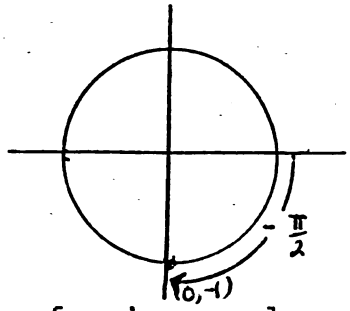


Example: If t=PI/2 (PI=3.141592654..), then you're at the top of the circle at (0,1). (The circumference of the unit circle is 2*PI*1=2*PI. The top is 1/4 of

the way around--1/4 of 2*PI is PI/2)

So, cos(PI/2)=0 and sin(PI/2)=1

Example: if t= -PI/2, then you're at the bottom of the circle at (0,-1). Thus cos(-PI/2)=0 and sin(-PI/2)=-1.



If you want to see some typical values of sine and cosine functions, write a short program:

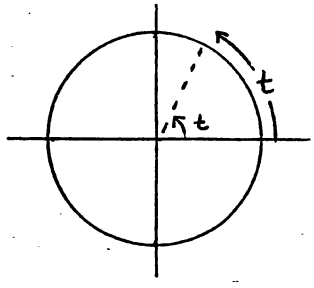
```
PRINT "T","COS(T)","SIN(T)"
FOR t=0 TO 2*PI STEP .1
PRINT t,COS(t),SIN(t)
NEXT t
```

To see that these values are indeed on a circle, RUN this following program:

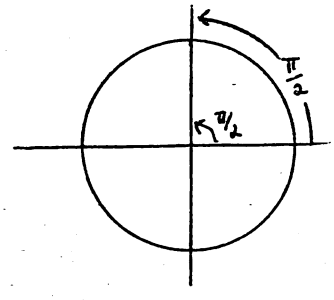
```
FOR t=0 TO 2*PI STEP .1
LET x=COS(t)
LET y=SIN(t)
LET xv=50*x+50
LET yv=-50*y+50
PLOT xv,yv
NEXT t
```

To understand the computation of the variables xv and yv you must realize I'm using a window of [-1,1]x[-1,1] and a viewport of [0;100]x[0;100]. Substitute these into the windowing function and you will get the corresponding equations for xv and yv.

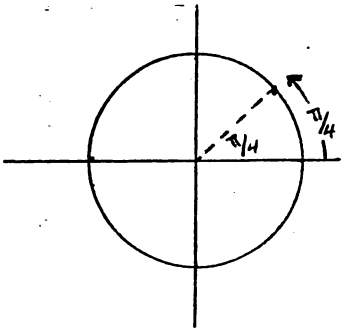
RADIAN MEASURE OF AN ANGLE



The numbers t are technically measurements along the circumference of the unit circle. However, it is common to use these same numbers as the measure of the central angle.(See figure to side of page) This type of angle measurement is called RADIAN measure(often abbreviated RAD). Thus a RAD measure of PI/2 is equivalent to 90 degrees, PI(half



way around the circle) is equivalent to 180 degrees, etc. I think you see why we have to tell a calculator which method of angle measurement is being used. If you have a calculator, put it in RAD mode, then find cos(PI/4). Now, put it in DEG mode, and find cos(45). The answers should be the same since PI/4 is equivalent to 45 degrees.



GFA BASIC only allows the Radian method of measuring angles for it's trigonometric functions. This is not true for all versions of BASIC.

SOME TRIG. FACTS AND RELATIONSHIPS

Since cos(t) and sin(t) are points on the unit circle, it is "obvious" that

$-1 \le \cos(t) \le 1$ and $-1 \le \sin(t) \le 1$ Also since $x^2 + y^2 = 1$ for any point on the unit circle, you get

$(\cos(t))^2 + (\sin(t))^2 = 1$ or more usually as

$$\cos^2(t) + \sin^2(t) = 1$$

This last relationship is often referred to as the

fundamental trig. identity

Example: If sin(A)=3/4 then the cos(A) can be computed as follows:

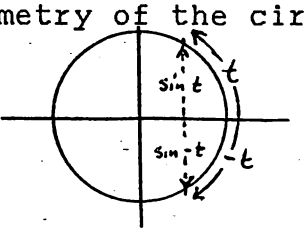
$\cos^2(A)+\sin^2(A)=1$
$\cos^2(A)+(3/4)^2= 1$
$\cos^2(A)=1-9/16=7/16$
$\cos(A)=\pm \sqrt{7/4}$.

Whether you pick the plus or minus sign for your answer depends on which quadrant of the circle A is in.

The following relations come from the symmetry of the circle:

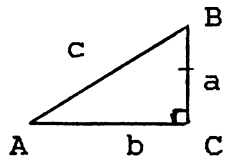
$$\cos(-t)=\cos(t) \quad \text{and}$$

$$\sin(-t)= -\sin(t)$$



AN ALTERNATE DEFINITION OF TRIG FUNCTIONS

The following definition is often referred to as the "right triangle" definition. Picture a right triangle as below:

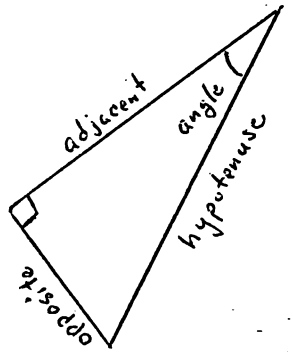


DEFINITION $\cos(A)=b/c$ $\sin(A)=a/c$

Sometimes these are memorized in the following way:

cos(angle)=adjacent/hypotenuse and

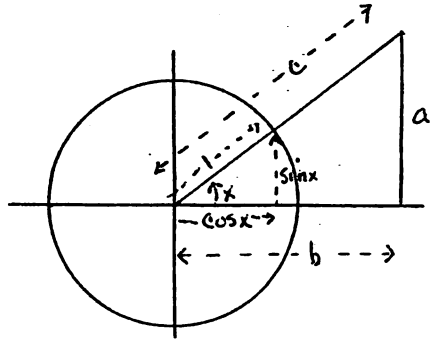
sin(angle)=opposite/hypotenuse



This allows for more flexiblity since many times the triangle isn't oriented exactly like the picture and different labels might be used for the sides.

Note that in the right triangle definition , it is easy to see that sin(A)=cos(B) and cos(A)=sin(B) or since B=90-A, sin(A)=cos(90-A) and cos(A)=sin(90-A) (we are thinking of the angle measured in degrees here. If we were using RAD measure, the same identities would be cos(A)=sin(PI/2-A), etc.)

Note that since the sides are always less than the hypotenuse, that the sine and cosine are in general less than 1(as we noted in circular definition). If you look at the picture to the right you should be able to see the reason why the two definitions basically give you the same numbers--just keep in mind that similar triangles have the same ratio for corresponding sides.



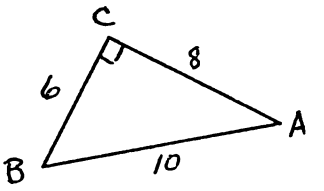
You can get the fundamental trig identity from this definition also.

$$\cos^2(A)+\sin^2(A) = (b/c)^2 + (a/c)^2 =$$

$$\frac{b^2}{c^2} + \frac{a^2}{c^2} = \frac{a^2+b^2}{c^2} = \frac{c^2}{c^2} = 1$$

problems

1. Look at the diagram to the side of this problem. Answer the following questions:



(a) sin(A)= ? (b) cos(B)= ? (c) sin(B)= ?

(d) $\sin^2(A)+ \cos^2(A)$= ?

2. Use your calculator(or computer) to find the values of the following:

(a) cos(30 degrees)=? (b) sin(70 degrees)=?

(c) sin(2.45)=?(use RAD mode) (d) cos(-3.45)=?

(e) cos(150 degrees)=?

3. Suppose a right triangle has the hypotenuse of 20 feet and one of the angles is 35 degrees. What is the length of the leg opposite the 35 degree angle?

SOME OTHER DEFINITIONS AND RELATIONSHIPS

Now that you know what we're talking about, let's delve into a few more things. What we're doing now won't really depend on which definition we use for the basic two trig functions.

There are four(4) more trig functions that can be defined:

tan(t) = sin(t)/cos(t) (tangent function)

cot(t) = 1/tan(t) (cotangent function)

sec(t) = 1/cos(t) (secant function)

csc(t) = 1/sin(t) (cosecant function)

There are two more fundamental trig identities involving these new functions.

$1 + \tan^2(t) = \sec^2(t)$

$1 + \cot^2(t) = \csc^2(t)$

(Can you establish these from our definitions and the original fundamental trig identity??)

There are literally hundreds of relationships between these six trig identities. Pick up a trig book to see a few of these. Most are not to be memorized, but are just drill type problems to increase your familiarity in using the basic definitions and fundamental identities.

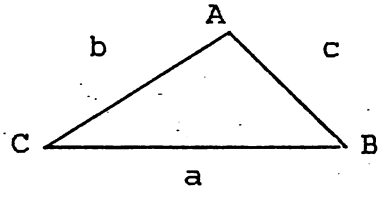
A couple of identities that we will need later are these:

** cos(x+y)=cos(x)cos(y)-sin(x)sin(y) **

** sin(x+y)=sin(x)cos(y)+cos(x)sin(y) **

These are relatively hard to derive. Drag out that book I told to to get!! Actually, only one of those is hard to get, the other comes relatively easy from the first one.

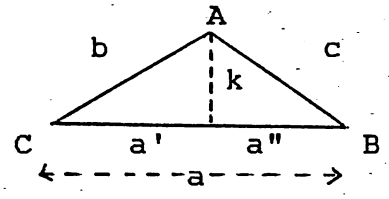
A very useful identity that involves non-right triangles is the following:



$$c^2 = a^2 + b^2 - 2ab*\cos(C)$$

It's kind of a modified pythagorean theorem with a "fudge-factor". Note that if C=90 degrees, we do get the Pythagorean theorem.(cos(90°)=0)

Proof:



$c^2 = k^2 + a"^2$ and $\sin(C)=k/b$ and $\cos(C)=a'/b$

$c^2 = b^2 \sin^2(C) + (a-a')^2$

$c^2 = b^2 \sin^2(C) + (a-b\cos(C))^2$

$c^2 = b^2 \sin^2(C) + a^2 - 2ab\cos(C) + b^2 \cos^2(C)$

$c^2 = a^2 + b^2 (\sin^2(C) + \cos^2(C)) - 2ab\cos(C)$

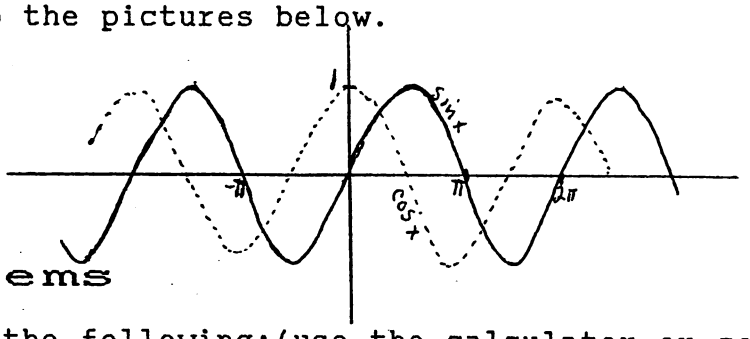
$c^2 = a^2 + b^2 - 2ab*\cos(C)$ **

In using this theorem, the actual letters are not important, but the relative position of the sides and the angle is the important thing. In that identity, the angle C and the side c are "opposite each other, and the a and b are the other two sides. Thus $a^2 = b^2 + c^2 - 2bc*\cos(A)$ would be an equally valid relationship.

Well, I'll end this "review"(?). You must remember that this is traditionally at least a semester high-school course. I've only tried to hit the high points as it relates to this book. To become competent in the use of trig, practice with a trig textbook is advised.

GRAPHS OF THE TRIG FUNCTIONS

"Crank up" your graphing program from the previous chapter and insert the SIN and COS functions and see what their graphs look like. I might suggest a window of [-12,12]x[-2,2] to see a good representative sample of the functions. Your graphs should look similar to the pictures below.



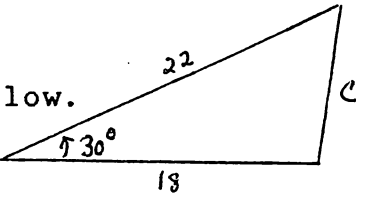
problems

1. Find the following:(use the calculator or computer along with the definitions given earlier)

(a) tan(67 degrees) (b) sec(35 degrees)

(c) csc(1.44) (d) cot(.44) (e) sec(-2.4)

2. Find the length of side c on the figure to the below.



3. Find a formula for these expressions:(hint: 2x=x+x)

(a) sin(2x) (b) cos(2x) (c) tan(2x)

4. Show that $\cos^2(x) - \sin^2(x) = 2\cos^2(x) - 1$.

5. If you know the sin(A)=1/2 and you know angle A is between 0 and 90°, what is the value of cos(A)? What if the angle is between 90 and 180 degrees?

6. If you know the sin(A) = 1/2 and the cos(B) = 2/3, what is the sin(A+B) and cos(A+B)?(hint: Figure out what cos(A) and sin(B) must be and use formulas for sine and cosine of sum of angles as given on page 8.)

chapter six

PARAMETRIC AND POLAR EQUATIONS

The following topics will give you a more in-depth look at different ways that curves can be generated from functions. We will first look closer at the general concept of parametric equations.

PARAMETRIC EQUATIONS

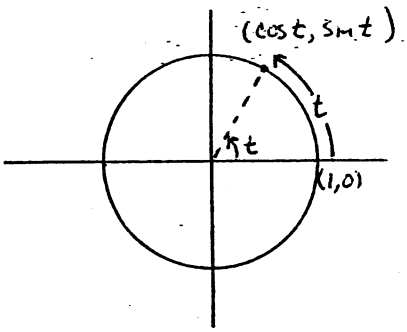
We've already met these when we looked at an alternate form for an equation of a line. We'll now generalize the concept and look at some different parametric equations.

DEFINITION: Let I=[a,b]. Let f and g be Real-valued functions on I. A curve is a function $F:I\to R^2$ defined by $F((t)=(f(t),g(t))$, $t\varepsilon I$. $x=f(t)$ and $y=g(t)$ are called the parametric equations of the curve. t is called the parameter.

Earlier, parametric equations of a line between two points (x_1, y_1) and (x_2, y_2) were developed. Namely:

$x=x_1+(x_2-x_1)t$ and $y=y_1+(y_2-y_2)t$, $t\varepsilon [0,1]$.

In trigonometry the following picture is a common sight:(See to the side) $x=\cos(t)$, $y=\sin(t)$ $t \varepsilon [0,2\pi]$ can be considered to be parametric equations of the unit circle(centered at (0,0)).



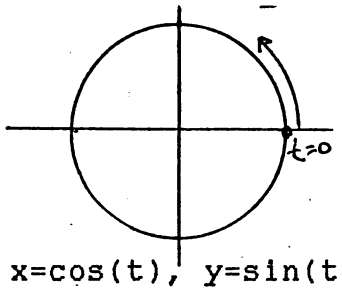
THEOREM: $x=r\cos(t)+h$ and $y=r\sin(t)+k$, $t \varepsilon [0,2\pi]$ can be considered to be parametric equations pf a circle of radius r and center (h,k).

Proof:
$x=r\cos(t)+h$ and $y=r\sin(t)+k$
$x-h=r\cos(t)$ and $y-k=r\sin(t)$
$(x-h)^2 + (y-k)^2 = r^2\cos^2(t) + r^2\sin^2(t)$
$(x-h)^2 + (y-k)^2 = r^2(\cos^2(t) + \sin^2(t))$
$(x-h)^2 + (y-k)^2 = r^2$ ##

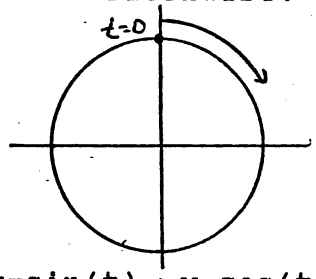
Often times different curves (different parametric equations) result in the same figure, (or a part of a standard figure) but

the figure is traced out in a different manner.

Example: $x=\sin(t)$ and $y=\cos(t)$, $t \in [0,2\pi]$ will still result in a circle ($x^2 + y^2 = 1$), but the "starting point" is (0,1) (at the top of the circle) and it is traced out clockwise.



$x=\cos(t)$, $y=\sin(t)$



$x=\sin(t)$, $y=\cos(t)$

Example: $x=1/\sqrt{1+t^2}$, $y=t/\sqrt{1+t^2}$ is the right-hand half of a circle.

Proof: $x^2+y^2= 1/(1/t^2) + t^2/(1+t^2) = (1+t^2)/(1+t^2) = 1$.

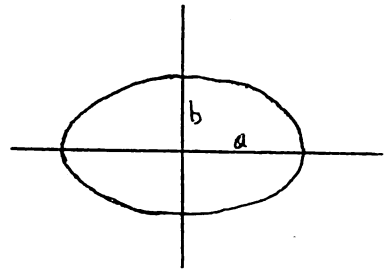
Since $x=1/\sqrt{1+t^2} > 0$ and $y=t/\sqrt{1+t^2}$ is greater than 0 if $t>0$ and less than 0 if $t<0$ we only get points in the 1st and 4th quadrants. ##

Convince yourself that no matter how big t gets, the half circle never quite is traced out.

Can you think of other parametric equations that give a curve that lies on a circle?

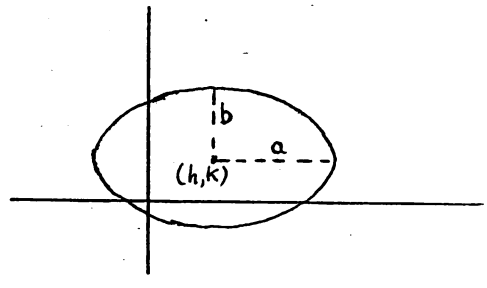
ELLIPSES AND PARABOLAS

A standard ellipse centered at (0,0) and with x and y intercepts of a and b, respectively is given by the following equation:



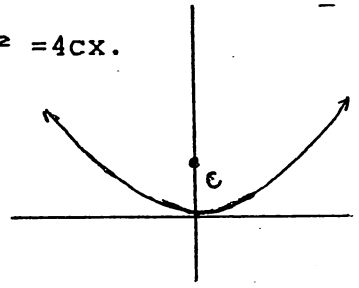
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

If that same ellipse is translated(moved) to a center of (h,k) the equation becomes:

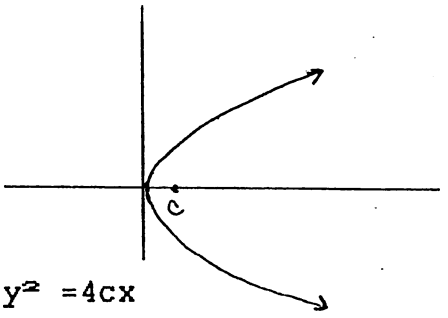


$$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$$

A standard parabola with its vertex at (0,0) where c is the distance to the focal point is given either by $x^2 = 4cy$ or $y^2 = 4cx$.

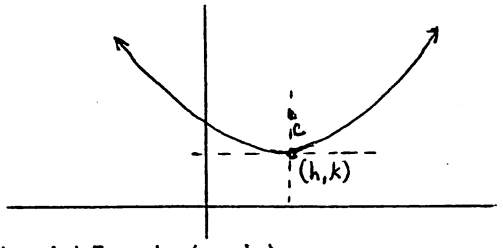


$x^2 = 4cy$

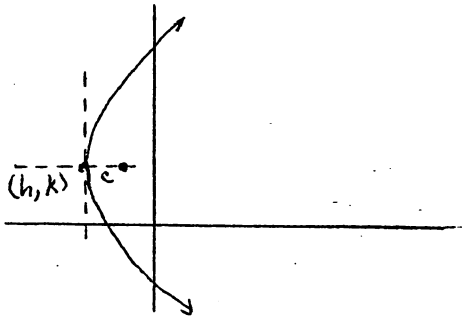


$y^2 = 4cx$

If these same parabolas are moved so that their vertex is at (h,k) then the equations are:



$(x-h)^2 = 4c(y-k)$



$(y-k)^2 = 4c(x-h)$

When either of these equations are multiplied out, they are easily recognizable they are parabolas by the fact that one variable is "squared" whereas the other one isn't.

Example: $y^2 + 3x + 6 = 0$ is a parabola(It's not in standard form however!) There are algebraic tricks to put such eqution in standard form, but I won't get into that here.(By the way the standard form for that above equation is $y^2 = 4(-3/4)(x+2)$-- vertex is at (-2,0) and the focal point is -3/4 a unit to the left of the vertex.)

Even though we can't spend the time going into all the aspects of graphing and forming equations of these curves(this is not an analytic geometry course), you should take the time to make up a few equations, pick a few values for x(or for y) and solve for the other variable and plot the points.

A usual parametric equation of an ellipse is

$x=a\cos(t)$ and $y=b\sin(t)$, $t\ [0,2\pi]$, ($a,b>0$)

To show this gives the standard equation, look at the following:

$x=a\cos(t)$ and $y=b\sin(t)$
$x/a=\cos(t)$ and $y/b=\sin(t)$
$(x/a)^2=\cos^2(t)$ and $(y/b)^2=\sin^2(t)$
$(x/a)^2+(y/b)^2=\cos^2(t)+\sin^2(t)$
$(x/a)^2+(y/b)^2=1$

This is an equation of a standard ellipse centered at the origin. Since cos(t) and sin(t) go between -1 and 1 as t goes once "around the circle", the x and y values go between -a and a and -b and b respectively.

As far as I know there is no "usual" parametric equation of a parabola. There are several that give parabolas: (a) $x=t$, $y=t^2$ or (b) $x=t^2$, $y=t$ or (c) $x=\sin(t)$, $y=\cos(2t)$ or ....
To see this last one, recall the trig identity:

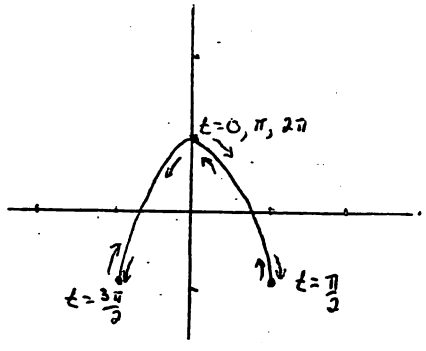
$\cos(x+y)=\cos(x)\cos(y)-\sin(x)\sin(y)$

If you replace both x and y with t, you get

$\cos(2t)=\cos^2(t)-\sin^2(t)$. Then using the fact that

$\sin^2(t)+\cos^2(t)=1$ which is equivalent to $\cos^2(t)=1-\sin^2(t)$, we get $\cos(2t)=1-\sin^2(t)-\sin^2(t)=1-2\sin^2(t)$

So, $y=\cos(2t)=1-2\sin^2(t)=1-2x^2$.
$y=1-2x^2$ is an equation of a parabola(not in standard form however). We are describing a curve that is only a part of a parabola since $-1\le \sin(t)=x \le 1$ and $-1\le \cos(2t)=y \le 1$. See the picture to the side of the page for a picture of this curve.



One reason a person doing graphics might wish to be aware of different ways to draw a particular curve, is that often the visual effect of seeing a curve drawn out is just as important as

the curve itself. The process of a curve being drawn is one of the most elementary forms of animation.

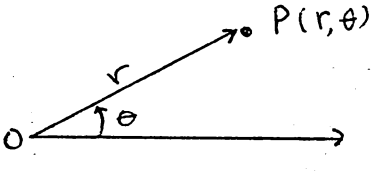
problems

Describe the figure these curves lie on. See if you can change these parametric equations to a "normal" x-y equation. Sketch the curves, describe the motion of a "bug" traveling along the curve(think of t representing time).

1. $x=2t+1$, $y=3t-4$, $t\varepsilon[1,5]$
2. $x=\cos(t^2)$, $y=\sin(t^2)$, $t\geq 0$
3. $x=\cos(t)$, $y=\cos(2t)$
4. $x=\sin(t)$, $y=\sin(2t)$. Just give its equation in terms of x and y. It doesn't have a common name that I'm aware of. Sketch the curve for $t \varepsilon [0,2\pi]$.
5. $x=|t|$, $y=t^2$
6. $x=|\cos(t)|$, $y=|\sin(t)|$
7. $x=t\cos(t)$, $y=t\sin(t)$. Don't try to put this in to an x-y equation. Just sketch the curve for $t\geq 0$.
8. Modify your graphing program to graph parametric equations. It shouldn't take much modification. Your graphing loop will depend on T, rather than X. You'll have to ask for input of the beginning and ending values of t.

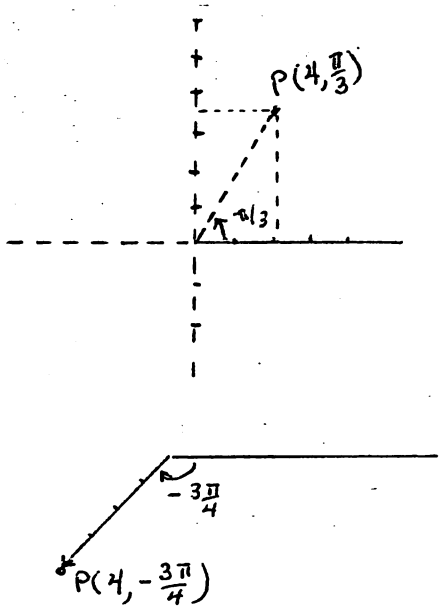
POLAR EQUATIONS

In the "real world" there are many ways to describe the position of a point in the plane. The most common way is to use the Cartesian Coordinate System(Rectangular Coordinate System). This is what has been used during our discussion of graphing so far in this book. Probably the second most common method is the Polar Coordinate System.

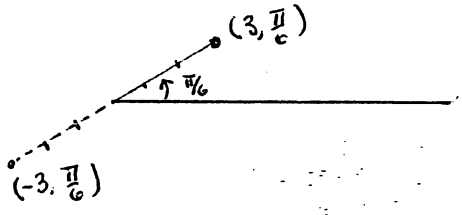


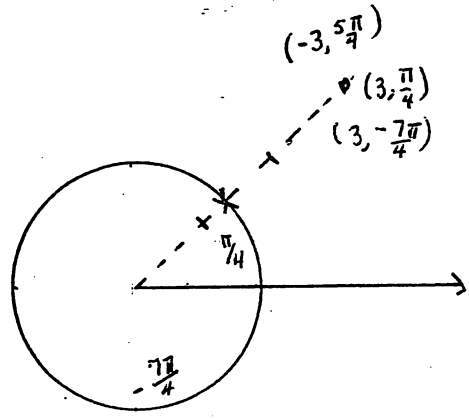
Look at the picture to the side of the page. The point P is referenced with respect to a fixed point O and a "half"

line from that point. It is usual to think of O to be the same as the origin in the rectangular coordinate system and the half line to be the positive x-axis. This way it is easy to compare coordinates of the same point using the two systems. Usually the r coordinate is a positive distance one moves after the angle has been determined. A positive angle gives a counter-clockwise rotation and a negative angle gives a clockwise motion.



If r is allowed to be negative, the convention is to consider the point to be r units on the ray that has been rotated $\theta + \pi$. This has the effect that for a negative value of r, go "backwards" from what you do if r was positive. One becomes immediately aware that there are MANY(infinite, in fact) ways to represent a point in polar coordinates. Consider a few, as you look at the diagram to the side. The coordinates $(3,\pi/4)$ or$(3,9\pi/4)$ or $(-3,5\pi/4)$ or $(-3,-3\pi/4)$ or..... all represent the same point.





This non-unique representation of points in the plane is a drawback of this particular coordinate system. However, the advantages when used in the right

situation completely outweighs that one disadvantage.

In the polar coordinate system, certain figures have extremely simple equations. Take a circle of radius 2, centered at O, for example. The equation is simply

$r=2$. (ie. $\{ (r,\theta): r=2 \}$

A straight line through the origin, oriented 30 degrees from the horizontal, has an equation $\theta = \pi/6$. (ie. $\{ (r,\theta): \theta=\pi/6 \}$

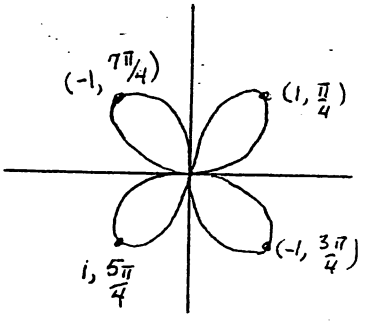
It is often the case that certain figures might be easy to represent in polar coordinates, but hard to represent in rectangular coordinates, and vice-versa.

You will find that most equations in the polar coordinate system are written in the form:

$$r = f(\theta)$$

such as: $r=\sin(2\theta)$, $r=\sin(\theta)+\cos(\theta)$, $r=1-2\cos(\theta)$, $r=\theta^2\cos(\theta)$

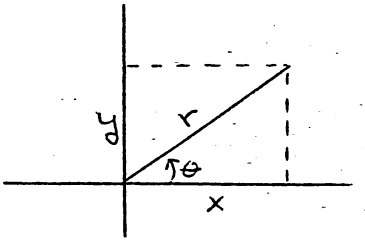
Often the trig. functions are incorporated as you can see from the examples.



Some very fascinating designs can be drawn using polar equations. For example, $r=\sin(2\theta)$ is a 4-leaf rose.

($\theta \in [0,2\pi]$)

Fortunately, for us computer users whose screen is basically set up on the rectangular coordinate system, there is an easy way to convert from polar coordinates to rectangular coordinates.



Basic trigonometry gives us that $x/r=\cos(\theta)$ and $y/r=\sin(\theta)$ or $x=r\cos(\theta)$ and $y=r\sin(\theta)$.

Now, even though in trig we always consider r to positive, <u>if</u> r

is negative the equations $x=r\cos(\theta)$, $y=r\sin(\theta)$ will still give us the right x and y coordinates. This is not too easy to verify in general, but examine this specific example:

$(4,\pi/6)$ or $(-4,7\pi/6)$

$x=4\cos(\pi/6)=3.464$ $y=4\sin(\pi/6)=2$

or

$x=-4\cos(7\pi/6)=3.464$ $y=-4\sin(7\pi/6)=2$

It turns out that if we consider r to be negative , we have to consider the angle $\theta+\pi$. The values of the sine and cosine functions of the different angle will compensate for the fact we are using a negative value for r. Note the following:

$$\begin{aligned} -r\cos(\theta+\pi) &= -r(\cos(\theta)\cos(\pi) - \sin(\theta)\sin(\pi)) = \\ &\quad -r(\cos(\theta)(-1) - \sin(\theta)(0)\ ) = \\ &\quad -r(-\cos(\theta) = \\ &\quad r\cos(\theta) \end{aligned}$$

Similarly $-r\sin(\theta+\pi)=r\sin(\theta)$. I'll leave that as an exercise.

So, if you've got the program "up and going" for graphing in rectangular coordinates, it can <u>easily</u> be converted to graph polar equations of the form $r=f(\theta)$. Do your FOR-NEXT loop on the values of θ , use your function to calculate a value for r, convert to rectangular coordinates by the equations above, and plot them as in your regular graphing program.

A simple outline of such a program might go as follows:

1. Enter viewport(or default to screen dimensions)
2. Enter window
3. Draw axis(optional)
4. Input beginning and ending values of θ. (Of course you'll have to use a different symbol in the program!)
5. In a FOR-NEXT loop, compute values of r, convert to rectangular coordinates, check to see if point is in window, convert to screen coordinates, PLOT the point.

As one final note, when you consider the equations that convert polar coordinates to rectangular coordinates, we can in essence think of polar equations being parametric equations with θ as the parameter!!

problems

1. Change the following polar equations to an equation in rectangular coordinates. Besides the two conversion formulas mentioned in the previous material, you might want to use

$x^2 + y^2 = r^2$ and/or $\tan(\theta)=y/x$.

(a) $r=2\sin(\theta)$ (HINT: mult. both sides by r)

(b) $r=2\sin(\theta)+4\cos(\theta)$

(c) $\tan(\theta)=4$ (d) $r^2=\sec(2\theta)$

(e) $r=2\sec(\theta)$ (f) $r=1/(1-\cos(\theta))$

2. Sketch the graphs of some of these equations in problem 1 or some of the other equations in this chapter. You may want to use a graphing program to help you sketch the results.

3. Convert these polar coord. points to rectangular coordinates.

(a) $(3,30°)$ (b) $(-4,57°)$ (c) $(2,\pi/5)$ (d) $(-3.2,-.47)$

4. Convert these rectangular coord. to polar coordinates. Give two equivalent polar coordinates for each.

(a) $(4,7)$ (b) $(-4,8)$ (c) $(-4,-7)$

5. Finish the proof that $-r\sin(\theta+\pi)=r\sin(\theta)$

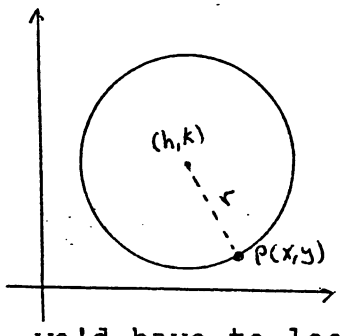
CIRCLES

I shall finish out this chapter with a discussion on graphing circles. Circles are common objects to be drawn on a computer screen. They are of some interest to us because there are 2 or 3 distinct ways to draw circles. Also, they are of interest because typically circles are slow to draw out because of the calculations that must be performed in most drawing routines. It is then of interest to see if one technique is faster than another.

THE OLD STANDBY

$(x-h)^2 + (y-k)^2 = r^2$ is the standard equation of a circle of radius r centered at (h,k). Or, solving for y, we get

$$y=\pm\sqrt{r^2 - (x-h)^2} + k$$



Since the computer can't handle this "± jazz", we'd have to look at this as two functions:

$$y=\sqrt{r^2 -(x-h)^2} +k \qquad y= -\sqrt{r^2 -(x-h)^2} +k$$

A sample routine might look like this:

(I'll not use the window routines in these short examples, hence the circles will graph themselves out "upside down" . You could consider the window to equal the viewport which is [0,639]x[0,199].)

```
LET r=30
LET h=100
LET k=60
FOR x=h-r TO h+r
  LET y=SQR(r^2-(x-h)^2)
  PLOT x,y+k
  PLOT x,-y+k
NEXT x
```

Just in this little example, we can experiment with the "speed" concept. Try replacing the first line in the loop with y=SQR(r*r-(x-h)*(x-h)) . I think you will see that simple

multiplication is performed much faster than exponentation.

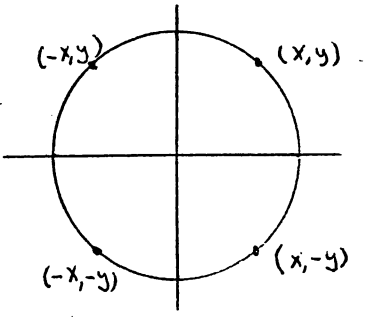
A more common way to view the circle at (h,k) is to think of it as a circle at (0,0) that's been <u>moved</u> to (h,k). In other words use $x^2 + y^2 = r^2$.

(or $y=\pm\sqrt{r^2 - x^2}$), then plot (x+h,y+k).

```
LET r=30
LET h=100
LET k=60
FOR x=-r TO r
  LET  y=SQR(r*r-x*x)  !  X*X IS FASTER THAN X^2
  PLOT x+h,y+k  ! TRANSLATES THE POINT BY AN AMOUNT H & K
  PLOT x+h,-y+k  !  DITTO
NEXT x
```

One could hasten the graphing by making use of more symmetry. Note the picture to the right:

```
LET r=30
LET h=100
LET k=60
FOR x=-r TO r
  LET  y=SQR(r*r-x*x)
  PLOT x+h,y+k
  PLOT x+h,-y+k
  PLOT -x+h,y+k
  PLOT -x+h,-y+k
NEXT x
```

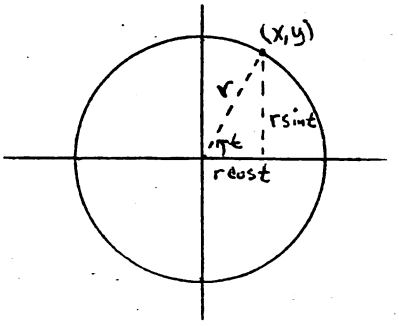


Adding h or k moved our points to where we wanted them. This concept of translating a picture to someplace else will be a major topic in a later chapter.

Can you write a program that draws a circle that takes even more advantage of the symmetry of the circle?

TRIG APPROACHES

From our previous discussion of the basis of trigonometry, we are familiar with the diagram to the right. Thus for a circle at the origin (0,0) the parametric equations



$$x=r\cos(t) \quad \text{and} \quad y=r\sin(t) \ , \ 0 \le t \le 2\pi$$

will do the job. To move the circle to a new origin (h,k) simply add h to x and k to y.

```
LET r=30
LET h=100
LET y=60
FOR t=0 TO 6.3 STEP 3.14/60  !  3 DEGREE INCREMENTS
  LET x=r*COS(t)
  LET y=r*SIN(t)
  PLOT x+h,y+k
NEXT t
```

(or replace the loop with:)

```
FOR t=0 TO 1.57 STEP 3.14/60  ! 0 TO 90 DEG STEP 3 DEG
  LET x=r*COS(t)
  LET t=r*SIN(t)
  PLOT x+h,y+k
  PLOT -x+h,y+k
  PLOT x+h,-y+k
  PLOT -x+h,-y+k
NEXT t
```

(Or replace the loop with:)

```
PLOT r*COS(0)+h,r*SIN(0)+k    ! PLOT FIRST POINT ON CIRCLE
FOR t=0 TO 3.14 STEP 3.14/60  ! 0 TO 360 DEG STEP 3 DEG
  LET x=r*COS(t)
  LET y=r*SIN(t)
  DRAW TO x+h,y+k   ! DRAW LINE TO NEXT POINT
NEXT t
```

Trig routines can be fairly slow because it is very time consuming to calculate values of trig functions.

A FASTER WAY

The two following formulas from trig will guide us toward more speed:

$$COS(A+B) = COS(A)COS(B)-SIN(A)SIN(B)$$

$$SIN(A+B) = SIN(A)COS(B)+COS(A)SIN(B)$$

Consider the diagram to the right:

Suppose we're trying to go from point (x_n, y_n) to the next point (x_{n+1}, y_{n+1}), incrementing the angle by a fixed amount dt. Using trig definitions for

coordinates of a point on the circle we get:

$$x_n = r\cos(t) \qquad y_n = r\sin(t)$$

$$x_{n+1} = r\cos(t+dt) \qquad y_{n+1} = r\sin(t+dt)$$

Let's now play with x_{n+1} and y_{n+1} by using our trig formulas.

$$x_{n+1} = r\cos(t+dt) = r[\ \cos(t)\cos(dt)-\sin(t)\sin(dt)\ ]$$

$$= r\cos(t)\cos(dt)-r\sin(t)\sin(dt)$$

$$= x_n \cos(dt) - y_n \sin(dt)$$

Similarly

$$y_{n+1} = r\sin(t+dt) = \text{you fill in}$$

$$= \text{the blanks}$$

$$= y_n \cos(dt) + x_n \sin(dt)$$

Since cos(dt) and sin(dt) would simply be constants, we easily get new points on our circle by using the previous point plotted and multiplying by constants.

```
LET r=30
LET h=100
LET k=60
LET dt=3.14/60 ! A 3 degree increment
LET s=SIN(dt)  ! A time-saver
LET c=COS(dt)
LET x=r        ! beginning value of x
LET y=0        ! beginning value of y
FOR t=0 TO 6.3 STEP dt
  PLOT x+h,y+k
  LET x1=x*c-y*s !Now compute the next point
  LET y1=y*c+x*s
  LET x=x1
  LET y=y1
NEXT t
```

The above routine draws out a nice looking circle clockwise (because our screen is upside down) in a very efficient manner. For further speed(sacrificing some elegance in animation) use the symmetry as outlined in the preceeding programs.

I'll finish by including(without comment) a very fast circle drawing routine. Note that there is no trig, square roots or even multiplcation.

```
LET r=30
LET h=100
LET k=60
LET phi=0
LET x=r
LET y=0
REPEAT
  PLOT x+h,y+k
  PLOT -x+h,y+k
  PLOT x+h,-y+k
  PLOT -x+h,-y+k
  PLOT y+h,x+k
  PLOT -y+h,x+k
  PLOT y+h,-x+k
  PLOT -y+h,-x+k
  LET phiy=phi+y+y+1
  LET phixy=phiy-x-x+1
  LET phi=phiy
  INC y
  IF ABS(phixy)<ABS(phiy) THEN
    phi=phixy
    DEC x
  ENDIF
UNTIL x<y
```

Why don't you see if you can put all of these programs together as one program to serve as a Circle demo program? I think you'd find it interesting to see how the circles draw out, one right after the other. Of course adjust the values of H and K so the circles don't all draw on top of each other.

chapter seven

CLIPPING

When condsidering a window (and its corresponding viewport) in doing graphics, it becomes important to determine what's in the window and what lies outside the window.

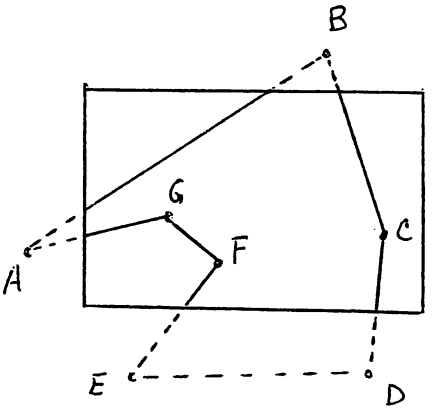
In displaying a complex shaped object, such as the one pictured to the right, several types of cases arise:

(1) Line segments whose endpoints are entirely contained in the window (line FG).

(2) Line segments whose endpoints are not in the window, but part of the line intersects the window (line AB).

(3) Line segments whose endpoints are not in the window and no part of the line is in the window (line DE).

(4) Line segments where one endpoint is in the window and other endpoint is not in the window (lines BC,CD,EF,AG).



Displaying only the portions of the object that intersect the window is called CLIPPING. It becomes the task in this chapter to develop an alogrithm and program that will perform this clipping process.

FIRST ALGORITHM

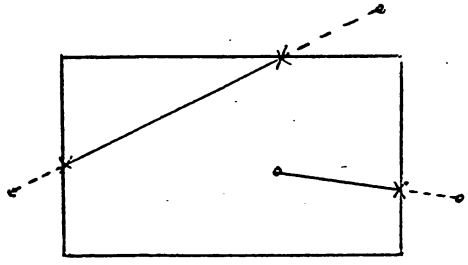
If one is graphing the lines, point by point, by using one of the earlier routines, the job is simple: compare the (x,y) coordinate of the point to be plotted with the minimum and maximum x and y values of the window. If the point lies in the window, plot it, otherwise don't plot it. This is often an unsatisfactory solution when working with a high-level language due to the

slowness of line plotting by plotting it point by point. However, in machine language this algorithm is extremely practical!

SECOND ALGORITHM

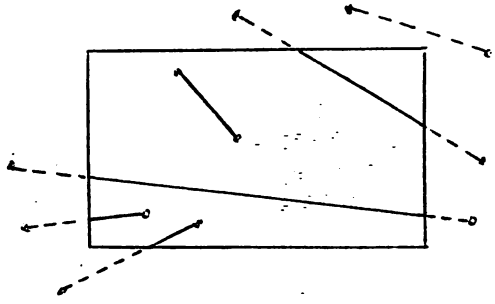
The algorithm we will develop assumes you will use the "built-in" drawing capability of the language.

(ie. DRAW x1,y1 TO x2,y2).



The algorithm, simply stated, is this: if one or both endpoints lie outside the window, determine the intersection points with the window. Then draw the line between the two points in (or on) the window.

The problem with this algorithm is the difficulty in writing a routine that covers all the possible positions of the line segments. There are so many possible orientations of the two points, and seemingly small changes result in different types of intersections with the window.

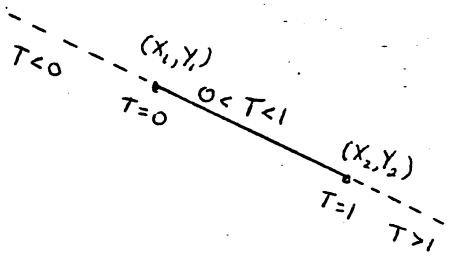


Sometimes you have no intersections; sometimes 1 intersection, either with a horizontal side or a vertical side; sometimes 2 intersections, both with vertical sides, or perhaps both with horizontal sides, or one with a vertical side and one with a horizontal side. Trying to write a routine that takes into account all of these possibilites, and yet is fairly quick and efficient becomes an interesting task.

The heart of the routine that will be shown you is the use of the parametric

equations of a line:

$X=X_1+(X_2-X_1)T$ and $Y=Y_1+(Y_2-Y_1)T$



where when T=0, we get (X_1,Y_1) and when T=1, we get (X_2,Y_2), and for any value of T between 0 and 1 we get points on the line segment between these two points.

We will also be interested in these equations solved for T:

$T=(X-X_1)/DX$ and $T=(Y-Y_1)/DY$

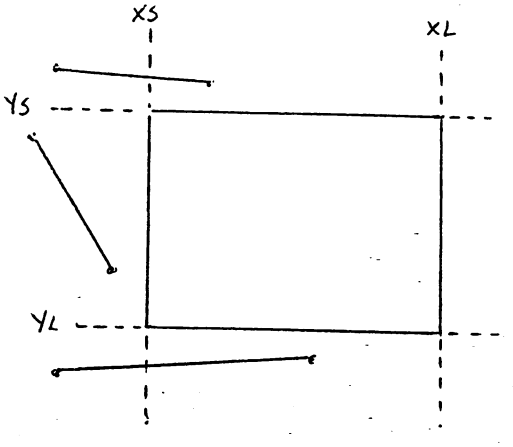
where $DX=(X_2-X_1)$ and $DY=(Y_2-Y_1)$

The program that follows (at end of chapter) was written by a student of mine the first time I taught a class in Math and Graphics. I had presented my routine, and he saw a better way to do the job. I invite you to build a better "mousetrap"!

I won't explain every part of the program as the ideas are duplicated for the different situations. Examine the listing that follows as the main parts are commented upon.

LINES beginning with IF (y1<ys AND y2<ys) OR (.....

This checks to see if both points are above, below, or to the sides of the viewport (The boundaries of the viewport are given by the variables xs,xl,ys,yl). If they are as shown in the figure to the right, then the subroutine RETURNs to main program witthout drawing a line.



LINES beginning with LET FLAG=0 .....

The routine assumes the vector is going from left to right, with (x1,y1) being the left-end point. The points are swapped if necessary. A variable called flag is set to 1 if a swap is made. The flag is checked before leaving this procedure(See last few

lines of this subroutine) and the points are "unswapped" if they had been swapped originally.

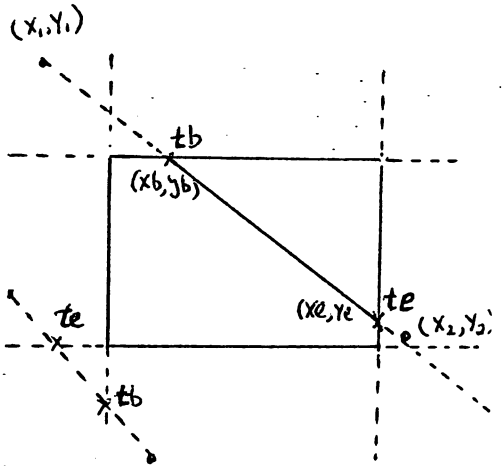
LINES beginning with Let dx=x2-x1 .....

The values of x2-x1 and y2-y1 are used repeatedly. Call these dx and dy respectively. The program then performs one of three actions if the line is vertical (dx=0), depending on whether dy is negative, zero, or positive.

LINES beginning with ELSE ! dx is not 0 .....

The program performs 1 of 3 actions depending if the line is uphill (dy<0--backwards from normal since the y-axis on the screen is upside-down), horizontal (dy=0), or downhill (dy>0).

Let's consider the THIRD "If statement"(If dy>0 THEN). The lines have an orientation as depicted to the right. Our explanation of what happens is typical of the process in the other "If statements".



Be sure to examine the figure to the right as this explanation proceeds. The figure shows the lines forming the viewport and the line segment itself being extended to show "intersections". Remember the line is being drawn left to right. A line that crosses the viewport will intersect the left side or top before it would hit either a right side or bottom (convince yourself this is not true for a line that misses the viewport).

To find the values of t for the left or top substitute xs or ys in for x or y in the parametric equations that were solved for t.(See top of page 3) Call one of the values t9 and the other tb. tb would be the larger of the two values(tb=MAX(tb,t9)). If tb is less than 0 then our starting point is within the boundaries of the viewport, so let tb=0(Recall that t=0 gives (x1,y1)--our starting point). Now we check the intersection with the right and bottom by letting x be xl and y be yl. We'll call these values t9 and te. We want te to be the smallest of these two values (te=MIN(te,t9)). If te>1 then the point was in the window, so we want te=1. Note in the picture to the right that "normally" tb is less than te if the line crosses the viewport. If tb is greater than te then the line misses the viewport. If te and tb are ok, then the x and y values associated with those values of t are calculated at the spot labeled draw_line at the end of the program. Then a line is drawn from (xb,yb) to (xe,ye).

It is hoped that you will examine the rest of the listing to thoroughly understand the logic. Don't be discouraged if you have trouble though. I found it hard to understand other clipping routines myself and finally wrote my own(which I understand!). I invite you to write your own routine also.

If you write your own routine or type in mine, be sure to save it as an ASCII file (SAVE,A option) so you can MERGE it with your other programs.

DEMO #1:(At end of chapter.)

In this little program, a series of diagonal lines are created in a FOR-NEXT loop. The program draws a viewport, then through the use of the clipping subroutine, draws only the portion of the lines that go through the viewport. If you type in the demo, be sure to MERGE in the clip_line procedure.

DEMO #2:(At end of chapter.)

This program draws a picture that is described in DATA statements. The figure is described in a window of [0,100]x[0,100]. Thus all the coordinates are positive. I take advantage of this and use negative numbers as flags in my Data lines.You can display this window or any part of that window on any viewport you wish. The program will draw the part of the figure that belongs in the viewport. In this program, the lines are drawn sequentially, hence the second point for the first line, becomes the first point of the next line to be drawn.

Be sure to try to find my initials on the door by picking the right window!

Be sure to look at the DATA lines to see how certain numbers in the data are used as flags to indicate that something else is happening besides drawing points. That's why those -1's and -9's are in the data list. The -1's indicate the pen is to be lifted and start a new sequence of lines. The -9's indicate the end of the data list.

progamming project

Use the Demo #2 program as a model and draw your own picture. Include some real small detail that won't show up unless the user picks a window that will magnify that portion of your picture. You might try a couple of levels of "smallness".

You might try to go beyound this example and try to add some color to your drawings. Try to set flags in your DATA that indicates you wish to change color (similar to the way you indicated to the program you wished to "lift your pencil" to start drawing in a new place.

```
PROCEDURE clip_line
  ' clipping subroutine--by Ken Hall--Highline CC--'84
  '  xs,xl,ys,yl--viewport dimensions
  ' x1,y1,c2,y2--endpoints of line
  LOCAL t9,tb,te,xb,yb,xe,ye,flag
  ' tb,yb,xb are values associated with the beginning point
  ' te,ye,xe are values associated with the ending point
  '
  IF (y1<ys AND y2<ys) OR (y1>yl AND y2>yl) OR (x1<xs AND x2<xs) OR (x1>xl AND x2>xl) THEN
    GOTO exit_sub
  ENDIF
  LET flag=0
  IF x2<x1 THEN
    SWAP x1,x2
    SWAP y1,y2
    LET flag=1
  ENDIF
  LET dx=x2-x1
  LET dy=y2-y1
  IF dx=0 THEN
    IF dy<0 THEN
      tb=MAX((yl-y1)/dy,0)
      te=MIN((ys-y1)/dy,1)
      GOTO draw_line
    ENDIF
    IF dy=0 THEN
      PLOT x1,y1
      GOTO exit_sub
    ENDIF
    IF dy>0 THEN
      tb=MAX((ys-y1)/dy,0)
      te=MIN((yl-y1)/dy,1)
      GOTO draw_line
    ENDIF
  ELSE    ! Dx is not 0
    IF dy<0 THEN
      t9=(xs-x1)/dx
      tb=(yl-y1)/dy
      tb=MAX(t9,tb)
      tb=MAX(tb,0)
      t9=(xl-x1)/dx
      te=(ys-y1)/dy
      te=MIN(te,t9)
      te=MIN(te,1)
      IF te<tb THEN
        GOTO exit_sub
      ENDIF
      GOTO draw_line
    ENDIF
    IF dy=0 THEN
      tb=MAX((xs-x1)/dx,0)
      te=MIN((xl-x1)/dx,1)
      GOTO draw_line
    ENDIF
    IF dy>0 THEN
      t9=(xs-x1)/dx
      tb=(ys-y1)/dy
      tb=MAX(tb,t9)
```

```
      tb=MAX(tb,0)
      t9=(x1-x1)/dx
      te=(y1-y1)/dy
      te=MIN(te,t9)
      te=MIN(te,1)
      IF te<tb THEN
        GOTO exit_sub
      ENDIF
      GOTO draw_line
    ENDIF
  ENDIF
  ' ********************
  draw_line:
  xb=x1+tb*dx
  yb=y1+tb*dy
  xe=x1+te*dx
  ye=y1+te*dy
  DRAW xb,yb TO xe,ye
  ' ********************
  exit_sub:
  IF flag=1 THEN
    SWAP x1,x2
    SWAP y1,y2
  ENDIF
RETURN

'************************************************************
'   Demo #1 of Clip_line subroutine
INPUT "enter viewport dim.(xs,x1,ys,y1) ",xs,x1,ys,y1
BOX xs,ys,x1,y1
FOR z=-140 TO 200 STEP 3
  x1=0
  y1=140+z
  x2=500
  y2=z
  clip_line
NEXT z
'    HERE IS WHERE TO MERGE IN YOUR CLIPLINE SUBROUTINE
```

```
'*****************************************************************
'  Demo #2 of using Clip_line routine
'  Demo of using clipping routine
'  A house is drawn using data statements
'
' try window of 60,70,18,25 to see initials on door
'
INPUT "Enter viewport dimensions(ie._5,400,4,150) ",xs,xl,ys,yl
INPUT "enter window(ie. 0,100,0,100 ) ",a,b,c,d
BOX xs,ys,xl,yl
LET w1=b-a
LET w2=d-c
LET v1=xl-xs
LET v2=ys-yl
RESTORE homedata
DO
  READ x,y  ! pen is up, new line
  EXIT IF x=-9
  window_function
  LET x1=x
  LET y1=y
  DO
    READ x,y
    EXIT IF x=-1 OR x=-9
    window_function
    LET x2=x
    LET y2=y
    clip_line
    LET x1=x2
    LET y1=y2
  LOOP
  EXIT IF x=-9
LOOP
homedata:
DATA 20,50,20,10,80,10,80,50,20,50,30,70,70,70,80,50,20,50,-1,-1
DATA 30,30,30,20,50,20,50,30,30,30,-1,-1
DATA 60,10,60,25,70,25,70,10,-1,-1
DATA 45,70,45,85,55,85,55,70,-1,-1
DATA 45,74,55,74,-1,-1,45,78,55,78,-1,-1,45,82,55,82,-1,-1
DATA 47,85,47,82,-1,-1,53,85,53,82,-1,-1
DATA 46,82,46,78,-1,-1,50,82,50,78,-1,-1,54,82,54,78,-1,-1
DATA 49,78,49,74,-1,-1,51,78,51,74,-1,-1,54,78,54,74,-1,-1
DATA 47,74,47,70,-1,-1,53,74,53,70,-1,-1
DATA 62,20,62,22,68,22,68,20,62,20,-1,-1
DATA 63,21,63,20.3,64.5,20.3,64.5,21,63,21,63,21.7,64.2,21.7,64.2,21,-1,-1
DATA 65.5,20.3,65.5,21.7,-1,-1,65.5,21,67,21,-1,-1,67,21.7,67,20.3
DATA -9,-9
PROCEDURE window_function
  x=(x-a)/w1*v1+xs
  y=(y-c)/w2*v2+yl
RETURN
'   HERE IS WHERE TO MERGE IN YOUR CLIPLINE SUBROUTINE
```

****

chapter eight

MATRICES

Before we can go much further in the traditional topics in graphics, I need to acquaint you with a new(?) concept. This chapter will essentially be "pure" math--later chapters will connect the topic to graphics.

MATRICES

A matrix is a rectangular array of numbers. An mxn matrix is a matrix with m rows and n columns.

Examples:

$$A=\begin{bmatrix}2 & 3 & 4\\ 3 & 7 & 8\end{bmatrix}$$ is a 2x3 matrix

$$B=\begin{bmatrix}3 & 4 & 7\\ 4 & 9 & 9\\ 2 & 4 & 1\end{bmatrix}$$ is a 3x3 matrix

$$C=\begin{bmatrix}2 & 1\\ 4 & 7\\ 2 & 8\\ 3 & 3\end{bmatrix}$$ is a 4x2 matrix

D= [3 4 7 9 8] is a 1x5 matrix

A square matrix is a matrix with the same number of rows and columns. Matrix B above is a square matrix.

A matrix of this type $\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$ is called an identity matrix

The following notation is often used when writing a matrix, using letters to represent the entries in the matrix.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots\cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots\cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots\cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots\cdots & a_{mn} \end{bmatrix}$$

This subscript notation allows one to refer to matrices in general or to make definitions of particular matrices. For example an identity matrix I can be described as follows:

$$I = [a_{ij}] \quad \text{where} \quad \begin{cases} a_{ii} = 1 & \text{if } i=1,2,..n \\ a_{ij} = 0 & \text{if } i \text{ does not equal } jj \end{cases}$$

MATRICES IN BASIC

Most versions of BASIC allow you to establish a matrix as a means of storing numbers.

In a program, a line like DIM A(7,5) will reserve memory space for a matrix of 8 rows and 6 columns (or 7 rows and 5 columns in some BASICS.) Generally there is a row "zero" and a column "zero". The DIM A(7,5) statement establishes the largest row and column number, so depending on whether your computer allows a 0 row and column, you get either 7 rows, 5 columns or 8 rows, 6 columns.

A statement in your program like Let A(4,3)=5 puts the number 5 in the spot that is referenced by row 4, column 3.
A statement in your program like Let T=A(3,2) assigns to the variable T the value found in the matrix A at the spot referenced by row 3 and column 2.

Hence you can see that in BASIC when we write A(3,2), this is equivalent to refering to a_{32} .(Assuming $A=[a_{ij}]$)

WHAT ARE THEY GOOD FOR?

Matrices might be set up for any number of reasons. Often it is just a good way to organize data.

Examples:

	Warehouse A	B	C
Chairs	500	10	200
Sofas	50	40	20
Tables	300	30	0

A matrix filled with 1's and 0's can be used to represent alphabetical characters to be drawn on a screen.

A crude

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

representation of A

The 1 tells the computer to plot a point, 0 means don't plot a point.

Sometimes the matrix might represent a series of points or vectors that are used to draw a figure.

X	Y
20	30
5	4
2	5
1	7
6	89

A matrix might represent a series of simultaneous equations to be solved.

$$\begin{aligned} 3x + 4y + z &= 7 \\ x - y + 2z &= -1 \\ 2x + y &= 4 \end{aligned} \qquad \begin{bmatrix} 3 & 4 & 1 & 7 \\ 1 & -1 & 2 & -1 \\ 2 & 1 & 0 & 4 \end{bmatrix}$$

In this book we will primarily be using matrices to represent transformations of points in doing graphics in the plane and 3-dimensional space.

MATRIX OPERATIONS

Besides representing "things", matrices can be made to behave like mathematical objects in their own right. To do this we need to define some operations that are commonly performed.

ADDITION

Let $A=[a_{ij}]$ and $B=[b_{ij}]$ be m x n matrices.

Define $A+B=[a_{ij}+b_{ij}]$.

Example:
$$A=\begin{bmatrix}2 & 3 & 4\\ 3 & 1 & 7\end{bmatrix} \qquad B=\begin{bmatrix}-1 & 2 & 6\\ 3 & 2 & 4\end{bmatrix}$$

$$A+B=\begin{bmatrix}1 & 5 & 10\\ 6 & 3 & 11\end{bmatrix}$$

The matrix $O=[0]$ is called the additive identity matrix since it would be true that $A+O=A$.

SCALAR MULTIPLICATION

Let $A=[a_{ij}]$ and let t be any real number(t is called a scalar).

Define $tA=[ta_{ij}]$.

Example:
$$A=\begin{bmatrix}2 & 3 & 4\\ 1 & 5 & 7\\ 3 & 2 & 1\end{bmatrix}$$

Then
$$2A=\begin{bmatrix}4 & 6 & 8\\ 2 & 10 & 14\\ 6 & 4 & 2\end{bmatrix} \qquad -3A=\begin{bmatrix}-6 & -9 & -12\\ -3 & -15 & -21\\ -9 & -6 & -3\end{bmatrix}$$

These two operations should remind you of our operations for vectors. If you think about it, a vector in the plane is a simple 1x2 matrix. As you will see later, this way of looking at a vector will be important.

MATRIX MULTIPLICATION

Let $A=[a_{ij}]$ be an $m \times n$ matrix and $B=[b_{ij}]$ be an $n \times p$ matrix.

AB is the $m \times p$ matrix $[\sum_{k=1}^{n} a_{ik} b_{kj}]$.

In other words if $\bar{C}=[c_{ij}]=AB$, then $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$

As you will see after we go through a couple of examples the number in row i, column j is obtained by multiplying row i of A with column j of B, adding as we go.

Example:

$$A=\begin{bmatrix} 2 & 1 \\ 3 & 2 \\ 4 & 1 \end{bmatrix} \qquad B=\begin{bmatrix} 1 & 3 & 7 & 4 \\ 2 & 1 & 5 & 2 \end{bmatrix}$$

3x2 matrix   2x4 matrix

AB is the 3x4 matrix:

$$\begin{bmatrix} 2*1+1*2 & 2*3+1*1 & 2*7+1*5 & 2*4+1*2 \\ 3*1+2*2 & 3*3+2*1 & 3*7+2*5 & 3*4+2*2 \\ 4*1+1*2 & 4*3+1*1 & 4*7+1*5 & 4*4+1*2 \end{bmatrix} = \begin{bmatrix} 4 & 7 & 19 & 6 \\ 7 & 11 & 31 & 16 \\ 6 & 13 & 33 & 18 \end{bmatrix}$$

This may seem like a complicated procedure, but it actually is pretty easy to do with a little practice. I find my hands really getting into the action, using a finger on the left hand to go across a row of the first matrix and a finger on my right hand to go down a column in the second matrix. It even helps to "mutter" out loud!! Look again at our previous example:

$$\begin{bmatrix} 2 & 1 \\ 3 \rightarrow & 2 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & 7 \downarrow & 4 \\ 2 & 1 & 5 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 7 & 19 & 6 \\ 7 & 11 & \textcircled{31} & 16 \\ 6 & 13 & 33 & 18 \end{bmatrix}$$

The number 31 which was in row 2, column 3 was obtained by "multiplying row 2 by column 3".

Example: Take our previous warehouse matrix.

Suppose each chair has a value of $200.
Suppose each sofa has a value of $500.
Suppose each table has a value of $300.

$$\begin{array}{c} \\ \text{Chairs} \\ \text{Sofas} \\ \text{Tables} \end{array}\begin{array}{c} \begin{array}{ccc} \text{A} & \text{B} & \text{C} \end{array} \\ \begin{bmatrix} 500 & 10 & 200 \\ 50 & 40 & 20 \\ 300 & 30 & 0 \end{bmatrix} \end{array}$$

Arrange in a 1x3 matrix:

$$\begin{array}{c} \\ \text{Value} \end{array}\begin{array}{c} \begin{array}{ccc} \text{Chair} & \text{Sofa} & \text{Table} \end{array} \\ \begin{bmatrix} 200 & 500 & 300 \end{bmatrix} \end{array}$$

$$\begin{bmatrix} 200 & 500 & 300 \end{bmatrix} \begin{bmatrix} 500 & 10 & 200 \\ 50 & 40 & 20 \\ 300 & 30 & 0 \end{bmatrix} = \begin{bmatrix} 215{,}000 & 31{,}000 & 50{,}000 \end{bmatrix}$$

represents the value of the inventory in each warehouse.

Example: The system of equations $2x+3y+4z=10$, $5x+2y-z=3$, $2x+y+3z=4$ is often represented by

$$\begin{bmatrix} 2 & 3 & 4 \\ 5 & 2 & -1 \\ 2 & 1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 3 \\ 4 \end{bmatrix}$$

in linear algebra.

problems

1. Let $A=\begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix}$ $B=\begin{bmatrix} 2 & 1 \\ 3 & 5 \end{bmatrix}$ $C=\begin{bmatrix} 2 & 3 & -1 \\ 4 & 5 & 7 \end{bmatrix}$

(a) Find A+B; AB; BC; 5B; BA

(b) Is matrix multiplication commutative?(Do any of your results above confirm your answer?)

(c) Why isn't CB defined in this example?

2. Let $A=\begin{bmatrix} 3 & -4 & 2 \\ 5 & 6 & 2 \\ -1 & 2 & -3 \end{bmatrix}$ $B=\begin{bmatrix} 2 & 1 & 2 \\ 0 & 1 & 0 \\ 5 & 2 & 1 \end{bmatrix}$

AB=? BA=? A^2 =AA=?

3. The matrix -A is defined to be the same as -1A. Using A as defined in problem 2, find -A. Is there a "reasonable way" to define subtraction of matrices--do so if you think there is.

4. $A=\begin{bmatrix}3 & 7 & 2\\ 2 & 4 & 1\\ 2 & 7 & 2\end{bmatrix}$ $I=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$

(a) Find AI and IA. In words summarize your conclusions about the matrix I.

(b) Let $K=\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}$ Find AK and KA.

In words summarize your conclusions about the matrix K and its effect on other matrices.

(c) Let $N=\begin{bmatrix}-1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$ Find AN and NA.

In words summarize your conclusions about the matrix N and its effect on other matrices.

5. Can you describe any other possible uses for matrices?(You might think of games, maps, photographs...)

chapter nine

MATRICES AND TRANSFORMATIONS OF POINTS

A square matrix of the form $\begin{bmatrix} A & C \\ B & D \end{bmatrix}$ can be thought of as a transformation of points in the plane.

Note:

$$[x\ y]\begin{bmatrix} A & C \\ B & D \end{bmatrix} = [Ax+By \quad Cx+Dy]$$

We have transformed a point (x,y) (thought of as a 1x2 matrix) to a new point (Ax+By,Cx+Dy). Note that in this case our matrix that did the transformation could be thought of as a function from $R^2 \rightarrow R^2$.

It might be noted we could get the same result by writing:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} Ax+By \\ Cx+Dy \end{bmatrix}$$

I, personally, kind of prefer this second form. However in most of the literature about transformations, the first form seems to prevail. Just as an aside, note the first matrix and the second matrix are related by the fact that their rows and columns have been interchanged. If this happens, often the one matrix is called the transpose of the other matrix.

For a transformation $\begin{bmatrix} A & C \\ B & D \end{bmatrix}$, the origin is special:

it will be left unchanged. To see this note that

$$[0\ 0]\begin{bmatrix} A & C \\ B & D \end{bmatrix} = [0*A+0*B \quad 0*C+0*D] = [0\ 0]$$

Also transformations of this type "preserve" lines. That is, points on a line, get transformed to points on another line. To see this, imagine we have a line segment between (x_1, y_1) and

(x_2,y_2). (x_1,y_1) gets transformed into a point (x_1',y_1') and (x_2,y_2) becomes a point (x_2',y_2'). For example,

$$[x_1',y_1'] = [x_1 \quad y_1] \begin{bmatrix} A & C \\ B & D \end{bmatrix} = [Ax_1 + By_1 \quad Cx_1 + Dy_1]$$

Let's see if a point on a line between (x_1,y_1) and (x_2,y_2) becomes a point between (x_1',y_1') and (x_2',y_2').

$(x_1+(x_2-x_1)t, \quad y_1+(y_2-y_1)t)$, $0 \le t \le 1$ is a point between (x_1,y_1) and (x_2,y_2).

$$[\, x_1+(x_2-x_1)t \quad y_1+(y_2-y_1)t \,] \begin{bmatrix} A & C \\ B & D \end{bmatrix} =$$

$$[\, Ax_1+A(x_2-x_1)t+By_1+B(y_2-y_1)t \quad Cx_1+C(x_2-x_1)t+Dy_1+D(y_2-y_1)t \,] =$$

$$[(Ax_1+By_1)+((Ax_2+By_2)-(Ax_1+By_1))t \quad (Cx_1+Dy_1)+((Cx_2+Dy_2)-(Cx_1+Dy_1))t] =$$

$$= [x_1'+(x_2'-x_1')t \quad y_1'+(y_2'-y_1')t \,], \quad 0 \le t \le 1$$

These are, as you can see, the parametric equations of a point on the line segment determined by (x_1',y_1') and (x_2',y_2'). Such a transformation is called a linear transformation.

Since lines remain lines, this means other figures pretty much retain their shapes.(They may get distorted, however.) This is because you generally can approximate any curve with "short" line segments.

TYPES OF TRANSFORMATIONS

Scaling

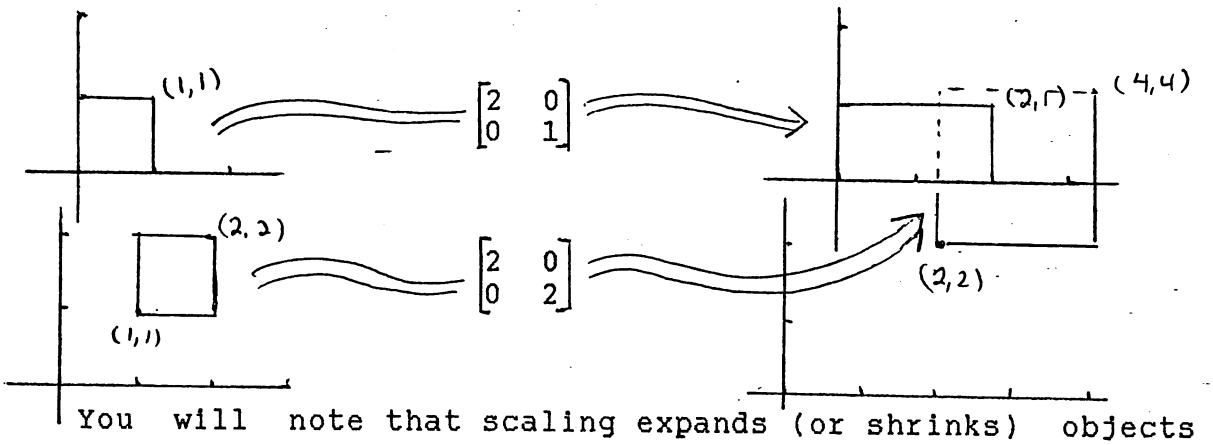
The size of an object may be controlled by a matrix of the type:

$$\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

$$[x \; y] \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} = [S_x \, x \quad S_y \, y]$$

Note that the x coordinate was multiplied by S_x and the y coordinate by S_y. This will give a "stretching" effect if S_x or

S_y >1 amd a "shrinking" effect if S_x or S_y are between 0 and 1. If $S_x=S_y$, then overall uniform scaling takes place.



You will note that scaling expands (or shrinks) objects away (toward) the origin. Remember the origin, if it is part of the figure you're scaling, does not move. These facts must be given consideration when trying for certain effects. Experiment by drawing simple objects and trying different combinations of positive numbers for S_x and S_y.

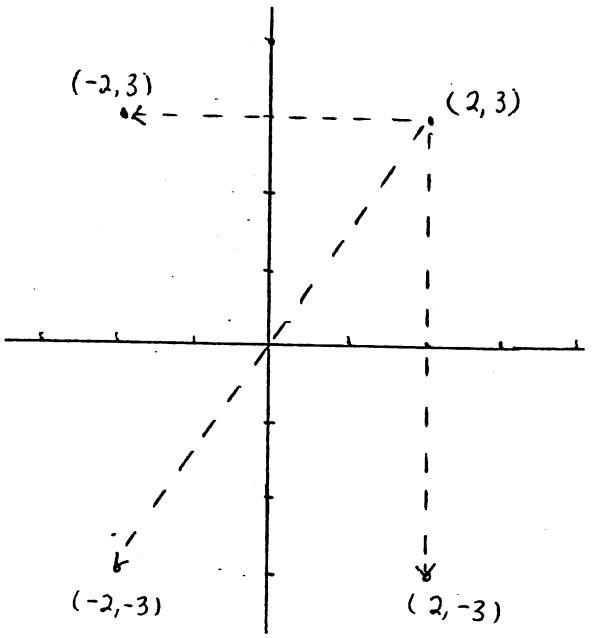
Reflection

These matrices cause symmetric movement of points.

About the x-axis: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

About the y-axis: $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

Through the origin: $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$



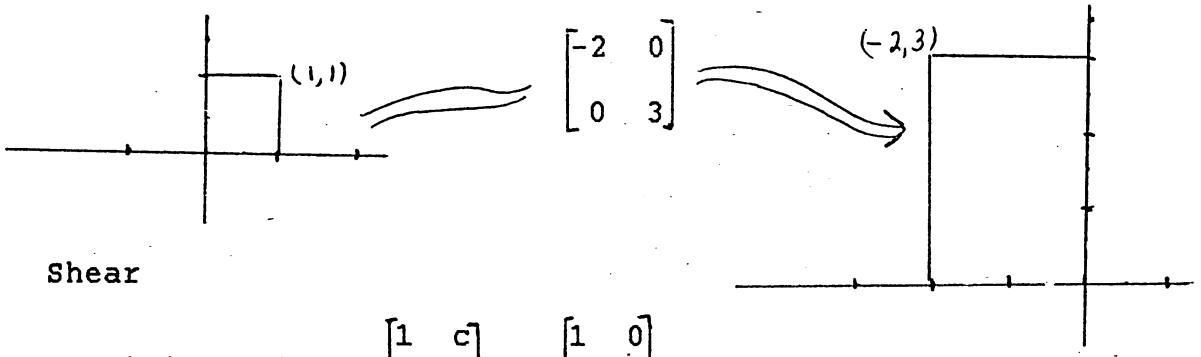
Examples: (a) $[2\ 3]\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = [2\ -3]$

(b) $[2\ 3]\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = [-2\ 3]$

(c) $[2\ 3]\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} = [-2\ -3]$

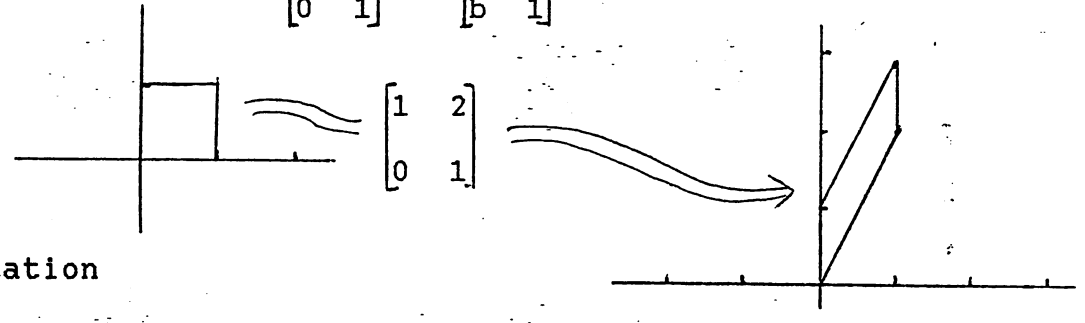
A matrix such as $\begin{bmatrix} -2 & 0 \\ 0 & 3 \end{bmatrix}$ is a combination of scaling and reflection.

Note that $\begin{bmatrix} -2 & 0 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$



Shear

A matrix such as $\begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix}$ gives a "shearing" effect.



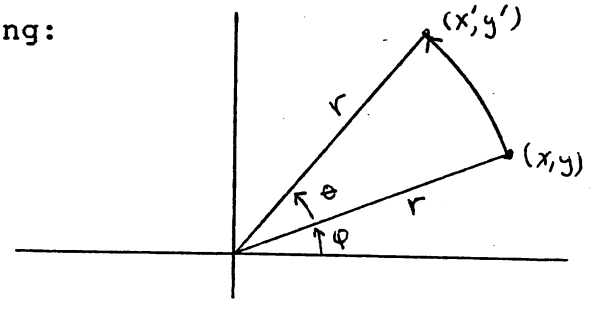
Rotation

A point is rotated in a counter-clockwise direction by the application of

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \quad \theta > 0$$

($\theta < 0$ gives rotation in clockwise direction.)

To see that this matrix gives the desired result, consider the following:



From trig we know that $x=r\cos\Phi$ and $y=r\sin\Phi$.

Also, looking at (x',y'): $x'=r\cos(\Phi+\theta)$ and $y'=r\sin(\Phi+\theta)$.

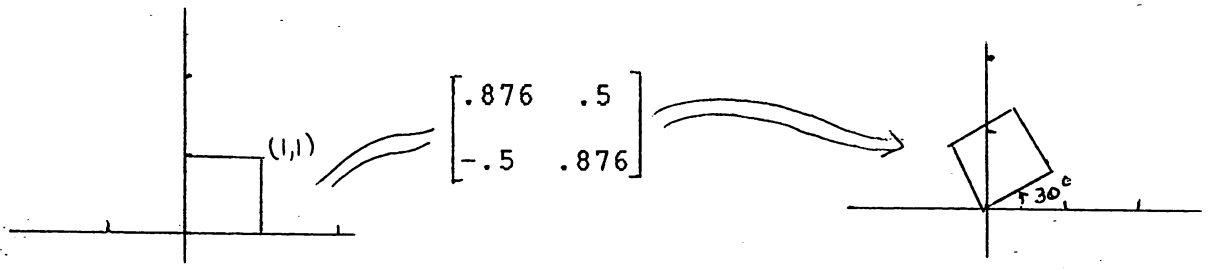
Applying the identities of $\cos(a+b)=\cos(a)\cos(b)-\sin(a)\sin(b)$

and $\sin(a+b)=\sin(a)\cos(b)+\cos(a)\sin(b)$

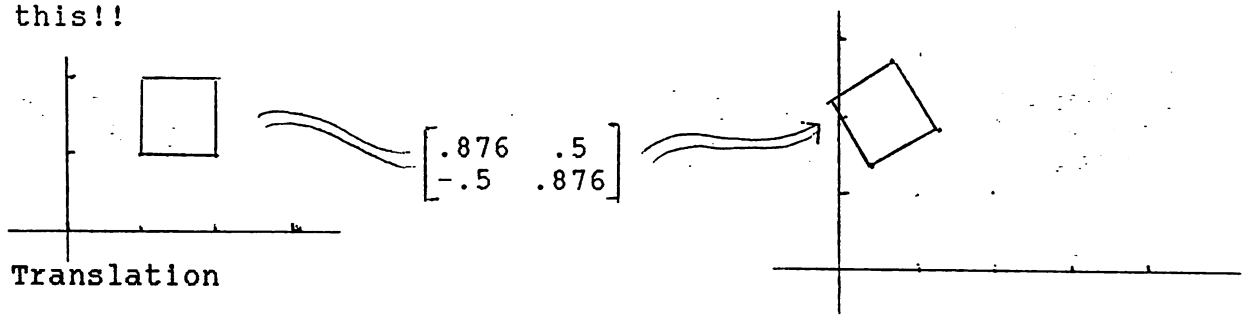
We get $x = r\cos(\Phi+\theta) = r(\cos\Phi\cos\theta - \sin\Phi\sin\theta)$
$= r\cos\Phi\cos\theta - r\sin\Phi\sin\theta$
$= x\cos\theta - y\sin\theta$

Similarly $y' = r\sin(\Phi+\theta) = r(\sin\Phi\cos\theta + \cos\Phi\sin\theta)$
$= r\sin\Phi\cos\theta + r\cos\Phi\sin\theta$
$= x\sin\theta + y\cos\theta$

Thus $[x'\ y'] = [x\ \ y]\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}$



Note that rotation matrices <u>always</u> assumes the origin is the center of rotation. This may cause a surprise if you forget this!!



Translation

Translation is a simple concept: simply add a constant to the x and y coordinates: $x' = x+h$ $y' = y+k$. However, you can not describe this transformation using 2x2 matrices. Rather than giving up on the idea of using matrices to describe this transformation, someone(?) came up with this ingenious idea: Define $(x,y,1)=(x,y)$. This is called the homogeneous coordinates of (x,y).

In place of our usual $\begin{bmatrix} A & C \\ B & D \end{bmatrix}$ use $\begin{bmatrix} A & C & 0 \\ B & D & 0 \\ 0 & 0 & 1 \end{bmatrix}$

For translation by h and k use $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ h & k & 1 \end{bmatrix}$

$$[x\ y]=[x\ y\ 1]\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ h & k & 1 \end{bmatrix} = [x+h \quad y+k \quad 1] = [x+h \quad y+k]$$

When combining transformations, deciding on which order things occur can be important. Look at the figures on the previous page. Suppose we take as our original object the square with the lower left point at the origin. The top figures show a rotation of 30 degrees. Now imagine a translation of h=1, k=1 (Not shown--sketch it in.) The bottom set of figures shows a translation of h=1, k=1 and <u>then</u> a rotation of 30 degrees. You should convince yourself that the result is <u>not</u> the same!!

The programs that follow utilize these transformation formulas. The first is a simple demo of transforming a simple object. You can not do multiple transformations (ie. progressively transforming an object).

The second program utilizes all of the "bells and whistles"--window, viewport, clipping, progressive transformations.

The third program is an "application" program that utilizes the ideas to draw "snowflakes".

The fourth program allows you to experiment with the transformation matrix in showing a familiar Northwest figure.

More about the 1st program:

This program assumes the viewport is the entire screen.

xC and yC are the center coordinates of the screen.

The window is basically [-xl/2,xl/2]x[-yl/2,yl/2]

The mapping formulas from the window to the viewport becomes very simple: x=x+xc;y=-y+yc

The program uses no clipping.

The program has the following simple outline:

(1) Initialize viewport, window, center of screen

(2) Draw x-y axis

(3) Get desired transformation

(4) Read Data, applying transformation

(5) Change to screen coordinates

(6) Draw the figure

problems

1. Give the matrix that would cause a vertical scaling of 5 and a horizontal scaling of 2.

2. Give the matrix that would cause an overall reduction in the size of a figure by a factor of 1/2.

3. Give the matrix that would cause a rotation of 90 degrees. Give a matrix that would cause a rotation of -45 degrees.

(Load up the last program(Seahawk) from your disk, and test your answers in problems 1-3--to see the original seahawk, make the matirx be the Scaling matrix with S_x and S_y both be 1.)

4. Give a matrix that would cause a translation of a figure 10 units to the right and 25 units down.

5. Give a matrix that would give the effect of a translation to the right 20 units, <u>then</u> a reflection about the y-axis.[Hint: consider multiplying matrices together.] Give a matrix that would reflect about the y-axis, <u>then</u> translate to the right by 20 units. Are these the same matrices?

```
'   simple demo of transformations
'          Simptran
'      Medium Resolution
LET xl=639
LET yl=159
LET xc=INT(xl/2)  ! CENTER OF SCREEN
LET yc=INT(yl/2)
'   DRAW IN AXIS
DRAW 0,yc TO xl,yc  ! Y AXIS
PRINT AT(77,12);INT(2*yc)
DRAW xc,0 TO xc,yl  ! X AXIS
PRINT AT(42,1);INT(yc)
read_and_plot("M")  ! plot the original figure (H=K=0)
DO
  '    TRANSLATION MENU
  REPEAT
    PRINT AT(1,21);"Choose    Move    Scale    Rotate    Quit (M,S,R
or Q)>>";
    LET choice$=UPPER$(INPUT$(1))
  UNTIL INSTR("MSRQ",choice$)>0
  PRINT choice$
  EXIT IF choice$="Q"
  IF choice$="M" THEN
    INPUT "Enter your translation--  H,K (return) >>",h,k
  ENDIF
  IF choice$="S" THEN
    INPUT "Enter the scale factors-- Sx,Sy (return) >>",sx,sy
  ENDIF
  IF choice$="R" THEN
    INPUT "Enter your rotation(degrees)--Theta (return) >>",theta
    LET theta=theta*PI/180  ! converts degrees to radians
  ENDIF
  RESTORE
  DATA 0,0,30,0,30,10,10,10,10,30,0,30,0,0,-9999,0
  COLOR 2
  read_and_plot(choice$)
  FOR i=21 TO 24
    PRINT AT(1,i);SPACE$(78)
  NEXT i
LOOP
PROCEDURE mapping_function
  '    window  is  [-2*Yc,2*Yc] x [-Yc,Yc]  a square would look approx.
square
  LET x=(x+2*yc)/(4*yc)*xl
  LET y=(y+yc)/(2*yc)*(-yl)+yl
RETURN
PROCEDURE read_and_plot(a$)
  '   READ AND PLOT THE DATA--Adapt this to the figure you want to
graph
  READ x,y
  translation_formulas(choice$)
  mapping_function
  DO
    PLOT x,y
    READ x,y
    EXIT IF x=-9999
    translation_formulas(a$)
    mapping_function
```

```
    DRAW  TO x,y
  LOOP
RETURN
PROCEDURE translation_formulas(a$)
  IF a$="M" THEN
    LET x=x+h
    LET y=y+k
  ENDIF
  IF a$="S" THEN
    LET x=x*sx
    LET y=y*sy
  ENDIF
  IF a$="R" THEN
    LET st=SIN(theta)
    LET ct=COS(theta)
    LET xx=x
    LET x=x*ct-y*st
    LET y=xx*st+y*ct
  ENDIF
RETURN
        ************************************************

'   Another demo of using transformations
'     Using Translation, Scaling, Rotation subroutines
'     Using clipping routine
'   The figure progressively changes with each succeeding transformation
'
view_window
draw_axis
'
'    LOAD UP ARRAY WITH THE PICTURE
DIM x(100),y(100)  ! Bigger than necessary
LET i=1
DATA 0,0,30,0,30,10,10,10,10,30,0,30,0,0,-1111,0,30,30,60,30,45,50,30,30,-9999,0
DO
  READ x(i),y(i)
  EXIT IF x(i)=-9999
  INC i
LOOP
'
'   GRAPH OUT ORIGINAL FIGURE
LET h=o
LET k=0
plot_figure("M")
COLOR 2
DO
  '   TRANSLATION MENU
  clear_textwindow
  REPEAT
    PRINT AT(1,21);"Choose    Move    Scale   Rotate   Quit (M,S,R
or Q)>>";
    LET choice$=UPPER$(INPUT$(1))
  UNTIL INSTR("MSRQ",choice$)>0
  PRINT choice$
  EXIT IF choice$="Q"
  IF choice$="M" THEN
    INPUT "Enter your translation-- H,K (return) >>",h,k
  ENDIF
```

```
  IF choice$="S" THEN
    INPUT "enter the scale factors-- Sx,Sy (return) >>",sx,sy
  ENDIF
  IF choice$="R" THEN
    INPUT "Enter your rotation(degrees)--Theta (return) >>",theta
    LET theta=theta*PI/180
  ENDIF
  plot_figure(choice$)
LOOP
PROCEDURE clear_textwindow
  FOR i=21 TO 24
    PRINT AT(1,i);SPACE$(70)
  NEXT i
  PRINT AT(1,21);
RETURN
PROCEDURE view_window
  clear_textwindow
  INPUT "Enter viewport Xs,Xl,Ys,Yl >> ",xs,xl,ys,yl
  INPUT "Enter window  A,B,C,D (Try -100,100,-100,100)>> ",a,b,c,d
  COLOR 1
  BOX xs,ys,xl,yl
RETURN
PROCEDURE draw_axis
  IF a<0 AND b>0 THEN
    screen_coordinates(0,c)
    PLOT xv,yv
    screen_coordinates(0,d)
    DRAW  TO xv,yv
  ENDIF
  IF c<0 AND d>0 THEN
    screen_coordinates(a,0)
    PLOT xv,yv
    screen_coordinates(b,0)
    DRAW  TO xv,yv
  ENDIF
RETURN
PROCEDURE screen_coordinates(x,y)
  LET xv=(x-a)/(b-a)*(xl-xs)+xs
  LET yv=(y-c)/(d-c)*(ys-yl)+yl
RETURN
PROCEDURE plot_figure(a$)
  LET i=1
  DO
    EXIT IF x(i)=-9999
    IF x(i)=-1111 THEN
      INC i
    ENDIF
    transform_point(x(i),y(i),a$)
    LET x(i)=xt
    LET y(i)=yt
    screen_coordinates(x(i),y(i))
    LET x1=xv
    LET y1=yv
    INC i
    DO
      EXIT IF x(i)=-1111 OR x(i)=-9999
      transform_point(x(i),y(i),a$)
      LET x(i)=xt
```

```
      LET y(i)=yt
      screen_coordinates(x(i),y(i))
      LET x2=xv
      LET y2=yv
      clip_line
      LET x1=x2
      LET y1=y2
      INC i
    LOOP
  LOOP
RETURN
PROCEDURE transform_point(x,y,a$)
  IF a$="M" THEN
    xt=x+h
    yt=y+k
  ENDIF
  IF a$="S" THEN
    LET xt=x*sx
    LET yt=y*sy
  ENDIF
  IF a$="R" THEN
    LET st=SIN(theta)
    LET ct=COS(theta)
    LET xt=x*ct-y*st
    LET yt=x*st+y*ct
  ENDIF
RETURN
'  merge in the clipping subroutine here
'  PROCEDURE clip_line

          ********************************************

'        Snowdemo
'    Another demo of using transformations
'    Using Translation, Scaling, Rotation subroutines
'    Using clipping routine
'  One leg of a snowflake is drawn, rest is drawn using
'  transformations.
'  Will use the same program outline as in previous program.
DIM x(13),y(13)
DO
  '  the data will be created "randomly"
  '
  view_window
  '
  '   LOAD UP ARRAY WITH THE PICTURE
  FOR i=1 TO 6
    LET x(i)=i-1
    LET y(i)=2*RND-1
  NEXT i
  LET x(7)=6
  LET y(7)=0
  '  reflect for rest of snowflake arm
  FOR i=6 DOWNTO 1
    LET sx=1
    LET sy=-1
    transform_point(x(i),y(i),"S")
    LET x(14-i)=xt
```

```
    LET y(14-i)=yt
  NEXT i
  '
  COLOR 2
  FOR angle=0 TO 300 STEP 60
    LET theta=angle*PI/180
    transform_point(x(1),x(2),"R")
    screen_coordinates(xt,yt)
    LET x1=xv
    LET y1=yv
    FOR i=2 TO 13
      transform_point(x(i),y(i),"R")
      screen_coordinates(xt,yt)
      LET x2=xv
      LET y2=yv
      clip_line
      LET x1=x2
      LET y1=y2
    NEXT i
  NEXT angle
LOOP
PROCEDURE clear_textwindow
  FOR i=21 TO 24
    PRINT AT(1,i);SPACE$(70)
  NEXT i
  PRINT AT(1,21);
RETURN
PROCEDURE view_window
  clear_textwindow
  INPUT "Enter viewport Xs,Xl,Ys,Yl >> ",xs,xl,ys,yl
  COLOR 1
  BOX xs,ys,xl,yl
  INPUT "Enter window - A,B,C,D(Try -10,10,-10,10) >> ",a,b,c,d
RETURN
PROCEDURE screen_coordinates(x,y)
  LET xv=(x-a)/(b-a)*(xl-xs)+xs
  LET yv=(y-c)/(d-c)*(ys-yl)+yl
RETURN
PROCEDURE transform_point(x,y,a$)
  IF a$="M" THEN
    xt=x+h
    yt=y+k
  ENDIF
  IF a$="S" THEN
    LET xt=x*sx
    LET yt=y*sy
  ENDIF
  IF a$="R" THEN
    LET st=SIN(theta)
    LET ct=COS(theta)
    LET xt=x*ct-y*st
    LET yt=x*st+y*ct
  ENDIF
RETURN
PROCEDURE clip_line
'  MERGE IN YOUR LINE CLIPPING SUBROUTINE
```

```
Program Seahawk
'    Seahawk
'  by Dick Plagge
'  Highline CC
' A PROGRAM THAT USES THE TRANSFORMATION MATRIX
'
DATA 0,0,15.3,0,15.2,1,15.2,2,15.5,3,15.8,4,16,4.4,16.1,4.7
DATA 14,8,0,8,0,0,-1111,0,0,10,14,10,15,10,15.5,11,16,11.3
DATA 17,12.1,18,12.7,19,13.2,20,13.6,21,14,22,14.1,23,14.2
DATA 23.5,14.22,24,14.2,25,14,26,13.7,27,13.3,28,12.8,29,12.3
DATA 29.5,12.25,30,12.3,30.5,12.6,31,13,31.6,13.8,31.8,13.5
DATA 31.5,13,31,12.4,30,12,29,11.85,28,12,27,12.4,26,12.8,25
DATA 13.1,24,13.25,23,13.3,22.5,13.33,22,13.3,21,13.2,20,13,19
DATA 12.6,18.5,12.2,18,11.8,17.5,11.4,17,10.8,16.5,5.5,17,4.8
DATA 16.5,3.9,16.2,3,16,2,16,1,16.1,0,39,0,40,-.1,41,-.3,42.2
DATA -.6,42.7,.4,42,.7,41,1,39,1.6,38,1.85,37,2.2,36,2.3
DATA 35,2.5,34,2.6,32,2.8,28,3.1,25,3.1,24,3,23,2.8,22,2.5
DATA 21,2,20.5,1.4,20.2,1,19.7,1,19.8,1.5,20,2,20.5,2.6
DATA 21,3.2,21.5,3.7,22,4,23,4.2,24,4.3,25,4.4,27,4.5,29
DATA 4.3,32,4,35,3.5,37,3,40,2,42.8,.9,43,1.25,43.2,2,43.6
DATA 4,43.8,5,43.9,5,43.7,8,43.5,9,42.9,10,43,11,41,11.5
DATA 40,11.8,39,12,35,12,34,10,33.2,8,33,7.7
DATA 32.6,7,32,6.3,31.5,5.85,31,5.7,30,5.5,28,5.6,26,5.8
DATA 25,6,24,6,23,5.8,22,5.55,21,5.2,20,4.5,19,3.5,18.5
DATA 2.5,18.2,2,18,1,19,1,19.4,2,20,3,20.5,3.6,21,4,22,4.5
DATA 23,4.6,26,5,28,4.9,30,4.7,33,4.2,35,3.9,38,3,41,2,43,1.25
DATA 42.8,.9,42.7,.4,42.2,-.6,41,-3.7,42,-3,43,-2.2,44,-1,45
DATA .5,46,3,46.5,4.5,46.8,6,46.9,8,46.8,9,46.6,10,46.4,11
DATA 46,12,45,13,44,13.6,43,14,24,14.2,35,14.2,33,16,32,16.6
DATA 30,17.6,29,17.8,28,18,0,18,0,10,-1111,0
DATA 30,10,28,11.3,26,12,24,12.4,23,12.6,22,12.6,21,12.5,20
DATA 12.2,19,11.7,18,11,17.3,10,17,9.3,16.9,9,16.8,8,16.9
DATA 6.5,17,6,17.5,5.4,18,6,18.5,6.3,19,6.5,20,6.8,21.5,7,22.5
DATA 6.9,23,6.8,24,6.65,25,6.6,26,6.65,27,6.8,27.5,7,28,7.3
DATA 28.7,8,29.4,9,30,10,-1111,0
DATA 28,10,27,10,26,10.4,25,10.9,24,11.3,23,11.4,22.5,11.3,22,11.2
DATA 21.5,11,21,10.7,20,10,18.4,10,18.5,9.5,20,9.5,20.5,9.8
DATA 21,10.3,22,10.8,23,11,24,10.9,25,10.3,26,10,27,9.6,28,9.6
DATA 28,10,-1111,0,28,9,28,9.4,27,9.3,26,8.8,25,8.2,24,7.8
DATA 23.5,7.7,23,7.8,22,8,21,8.5,20.5,8.8,20,9,28.5,9,18.5,8.6
DATA 20,8.6,20.5,8.5,21,8.3,21.5,8,22,7.7,23,7.5,23.5,7.4,24,7.5
DATA 24.5,7.6,25,7.8,25.5,8.2,26,8.5,26.5,8.7,27,8.8,27.5,8.9,28,9
DATA -1111,0,21,9,22,9,22,9.3,21,9.3,21,9,-1111,0
DATA25,9.5,24.95,9.9,24.8,10.3,24.6,10.6,24.3,10.8,23.9,10.95,23.5
DATA 11,23.1,10.95,22.8,10.8,22.4,10.6,22.2,10.3,22.05,9.9,22,9.5
DATA 22.05,9.1,22.2,8.7,22.4,8.4,22.8,8.2,23.1,8.05,23.5,8
DATA 23.9,8.05,24.3,8.2,24.6,8.4,24.8,8.7,24.95,9.1,24,9.5
DATA -9999,0
REPEAT
  FOR i=21 TO 24
    PRINT AT(1,i);SPACE$(70);
  NEXT i
  PRINT AT(1,21);
  PRINT "Enter entries of matrix   | A C |"
  PRINT "                          | B D |"
  INPUT "Enter A,B,C,D--try 5,0,0,2 as a start ",a,b,c,d
  COLOR 1
  DEFFILL 0,1
```

```
  PBOX 0,0,639,159
  COLOR 2
  RESTORE
  DO
    READ x,y
    matrix_trans
    PLOT x,y
    DO
      READ x,y
      EXIT IF x=-1111 OR x=-9999
      GOSUB matrix_trans
      DRAW  TO x,y
    LOOP
    EXIT IF x=-9999
  LOOP
UNTIL a=0 AND b=0 AND c=0 AND d=0
'
PROCEDURE matrix_trans
  LET x1=x
  LET y1=y
  LET x=a*x1+b*y1
  LET y=c*x1+d*y1
  LET x=x+200
  LET y=-1*y+100
RETURN
```

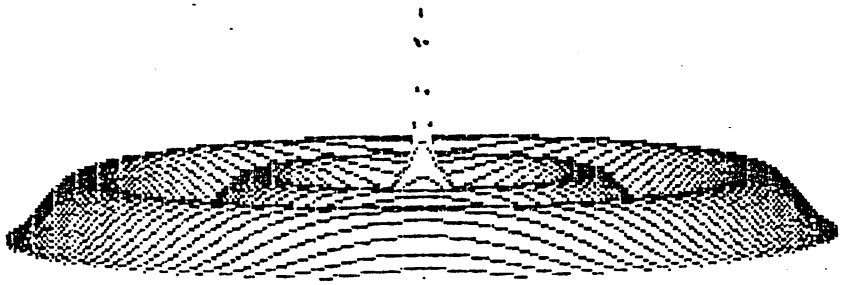
****

a programing assignment

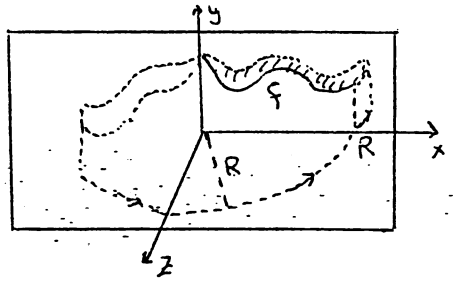
Modify the Seahawk program so you enter in the matrix that allows translation(ie. Homogeneous Coordinates).

chapter ten
SURFACES OF REVOLUTION

This chapter will ease us into the world of 3-dimension. A very striking figure is a three dimensional graph that is a symmetric about the origin. The figure below is a sample of such a graph.



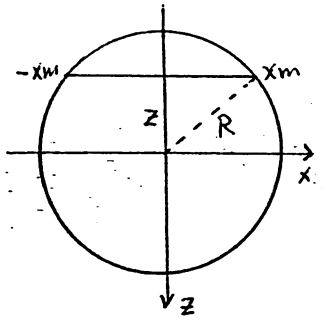
The figure is formed by considering the graph of a simple function in the x-y plane and then rotating that figure around the y-axis.(See the sketch below) This is called a surface of revolution.

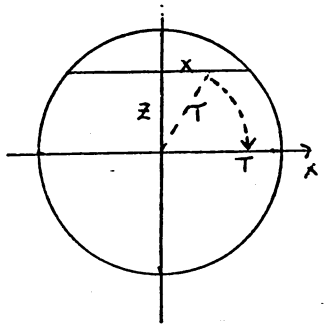


We will imagine a coordinate system in which the z-axis comes out of the screen toward the viewer. (We will discuss coordinate systems in much more detail in the next chapter.) I will draw the surface in cross-sections parallel to the x-y plane(screen) from back to front. Imagine we're looking down on the surface from the top.(See the two figures on the next page.) If R is the radius of revolution, then the z-values will go from -R to R.

Program to graph surface of revolution:

```
'          ThreeD
' A surface of revolution program
'
LET xl=639
LET yl=199
LET xc=INT((xl+1)/2)
LET yc=INT((yl+1)/2)
LET r=xc-5 ! make radius of rotation slightly smaller than 1/2 screen
FOR z=-r TO r STEP 10  ! picks cross-sections from back to front
  LET xm=SQR(r*r-z*z)
  FOR x=-xm TO xm  ! picks x values for each cross-section
    LET t=SQR(x*x+z*z) ! value to put into function
    LET y=10*COS(0.05*t)+300/(t*t+3) ! THE FUNCTION WE'RE PLOTTING
    '  WERE NOT USING A FANCY WINDOWING FUNCTION, THE VALUES OF
    '  X AND Y ARE BASICALLY SCREEN COORDINATES
    LET y=y-0.09*z  ! tips back edge up--corresponds to about 5 degrees
    LET x1=x+xc
    LET y=-1*y+yc
    IF y<yl AND y>0 THEN
      PLOT x1,y
      COLOR 0
      DRAW x1,y+1 TO x1,yl
      COLOR 1
    ENDIF
  NEXT x
NEXT z
```





The value of xm is the maximum x for that cross-section. The first top view shows the relation between z, r, xm. By the Pythagorean theorem, $xm=SQR(r^2-z^2)$.

The second top view shows an x picked on the cross-section. The distance t to that point (from the origin) is related to z and x . Again the Pythagorean theorem comes into action. This time as $t = SQR(x^2 + z^2)$.

Any point a distance of t from the center of rotation will have the same y-value, so consider t the value to be placed into the function, and compute f(t).

The sneaky part of the program is the way we do not show any hidden lines. Since we're drawing the surface from back to front, any cross-section in front will automatically block out "stuff" behind it. So the program plots the point, changes the color to the background color, then draws a line from below that point to the bottom of the screen. This effectively erases the previously drawn "stuff" that was lower than the plotted point. The color is then changed back to the plot color.

We now tip the back of the figure up and the front down for better viewing. If z is negative, then y-0.09x is greater than the original y since -.09*z would be positive. Similarly if z is positive, y-0.09z makes y smaller. The 0.09 gives about a 5 degree tip.

ANOTHER WAY

A second approach to graphing the same surface is to graph it from front to back. At each x-postion keep track of the largest y-value plotted and the smallest y-value plotted. Use two arrays (ub and lb--DIM ub(xl),lb(xl)) to keep track of this information. If a point is above or below a previously drawn point then plot it. See the next program for the implementation of this idea.

```
'           ThreeDee
'  Another surface of revolution program
'
LET xl=639
LET yl=199
DIM ub(xl),lb(xl)
ARRAYFILL ub(),-9999
ARRAYFILL lb(),9999
LET xc=INT((xl+1)/2)
LET yc=INT((yl+1)/2)
LET r=xc-5 ! make radius of rotation slightly smaller than 1/2 screen
FOR z=r TO -r STEP -10  ! picks cross-sections from front to back
  LET xm=INT(SQR(r*r-z*z))
  FOR x=-xm TO xm  ! picks x values for each cross-section
    LET t=SQR(x*x+z*z) ! value to put into function
    LET y=10*COS(0.05*t)+300/(t*t+3) ! THE FUNCTION WE'RE PLOTTING
    '  WERE NOT USING A FANCY WINDOWING FUNCTION, THE VALUES OF
    '  X AND Y ARE BASICALLY SCREEN COORDINATES
    LET y=y-0.09*z  ! tips back edge up--corresponds to about 5 degrees
    LET xl=x+xc
    LET y=-1*y+yc
    LET upper=ub(r+x)
    LET lower=lb(r+x)
    IF y<=upper AND y>=lower THEN
      ! do nothing
    ELSE
      ub(r+x)=MAX(upper,y)
      lb(r+x)=MIN(lower,y)
      IF y<yl AND y>0 THEN
        PLOT xl,y
      ENDIF
    ENDIF
  NEXT x
NEXT z
SGET picture$  !saves screen in the variable picture$
PRINT AT(1,22);
INPUT "Do you wish to print this picture? ",answer$
IF UPPER$(answer$)="Y" THEN
  PRINT AT(1,24);"Be sure printer is on.  Press ALTERNATE HELP"
  PRINT " when this message disapperars.  When done, press RETURN"
  PAUSE 250
  SPUT picture$  !puts picture back on screen
  INPUT "",a$ ! keeps program from ending
ENDIF
INPUT "Do you wish to save this picture? ",answer$
IF UPPER$(answer$)="Y" THEN
  INPUT "Enter name of file to save this under. ",name$
  SPUT picture$  ! put picture on screen(no writing!)
  BSAVE name$,XBIOS(2),32000 ! Xbios(2) gives address of screen
  '                          It takes 32000 bytes for a picture
ENDIF
'  The following shows how to put picture on screen from the file
' BLOAD name$,XBIOS(2) ! you might want an input to get the
'                       name of the file
```

The second method seems to be faster although both are very time-consuming. The first method uses a lot less memory since it is "expensive" byte-wise to store numbers in arrays. But what the heck--we got memory to burn these days!

Try other functions in the program!! You might have to experiment a bit to get the constants adjusted so you get a figure you like. You might consider increasing the step size on the outside loop as your experimenting, so you can get a rough idea quickly as to how your creation looks.

By the way, the variables "upper" and "lower" are upper and lower bounds of the values of y at each x-position. Since the screen coordinates are somewhat upside down, these are the reverse of their physical location on the screen.

Later, after we've discussed the process of graphing in 3-D more thorougly, I'll try to present a fairly simple program to graph general 3-D surfaces.

chapter eleven

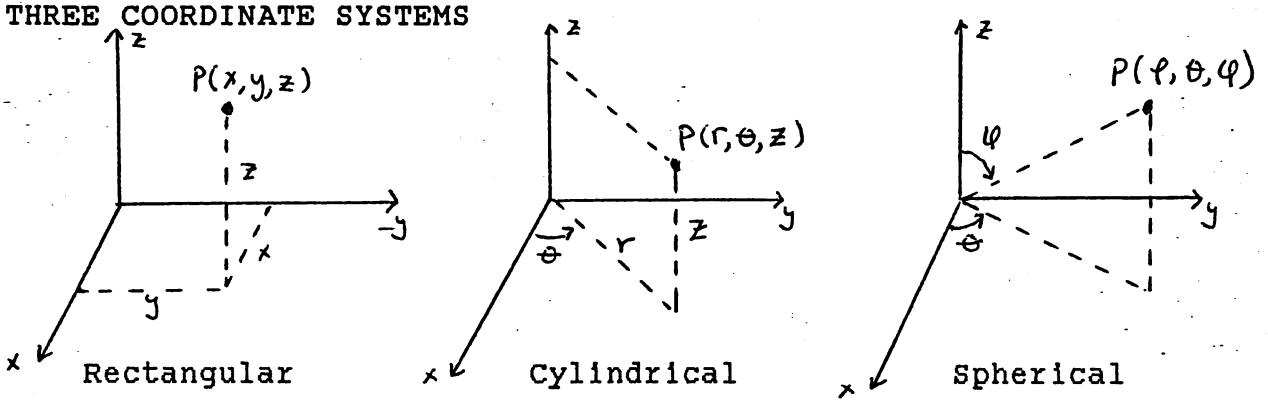
"THE REAL" REAL WORLD OF 3-D

The job at hand(and for the rest of this book) is to look at the three dimensional world. We need to figure out how to represent 3-D objects on the little two dimensional screen of the computer. Making the 3-D object look realistic can involve such diverse topics such as perspective, shading, color, hidden lines and surfaces, texture, and shadows. At this point, only the most sophisticated computers have the capability to represent all of these characteristics realistically. This book will stick to the most basic of 3-D graphics. Of course, our "old" topics of windows, viewports, and clipping are still relevant.

The mathematics needed to discuss all of these topics is much beyond a short book in graphics. Also, the equipment needed to demonstrate all of these topics is beyond the resources of the average person with a microcomputer.(However, the advent of the new cheap 16-bit computers with "super" high resolution graphics is putting this into reach.)

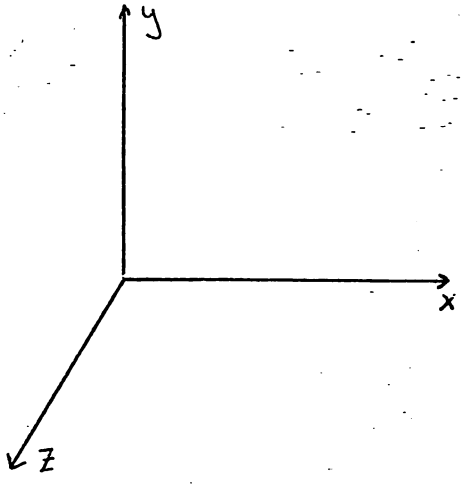
To start off, we need to be aware of how points in 3-D can be represented. Some of the standard coordinate systems will now be looked at.

THREE COORDINATE SYSTEMS



RECTANGULAR COORDINATE SYSTEM

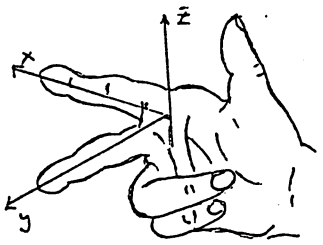
As you can see, our 3-D rectangular coordinate system is a natural extension of the 2-D cartesian coordinate system. Many formulas are just extensions of 2-D formulas. For example, the distance between two points (x_1,y_1) and (x_2,y_2) is given by $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$. In three dimensions, the distance between (x_1, y_1, z_1) and (x_2, y_2, z_2) is given by $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$. The orientation of the xyz-axis can vary from book to book, however the one shown above is the most common in mathematics books. However the one to the right is a common orientation for computer graphics. Note that the x-y axis are in normal position as in the two dimension case, but the z axis comes out toward us.(Out from the screen, if the screen represents the x-y plane.)



If you can imagine a stick model of the axis, other representations can be imagined by simply rotating it into different orientations. All are what called "right-handed" systems. The name stems from using the fingers of the right hand to represent the x-y axis and the thumb to represent the z-axis. There are a couple of ways to do this. Both, however, strain my

artistic abilities to draw pictures.

or

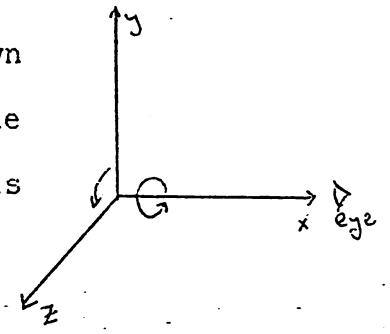


Imagine pointing all your fingers toward the x-axis, then curl your fingers into a fist toward the y-axis. In doing this, your thumb will point in the direction of the z-axis.

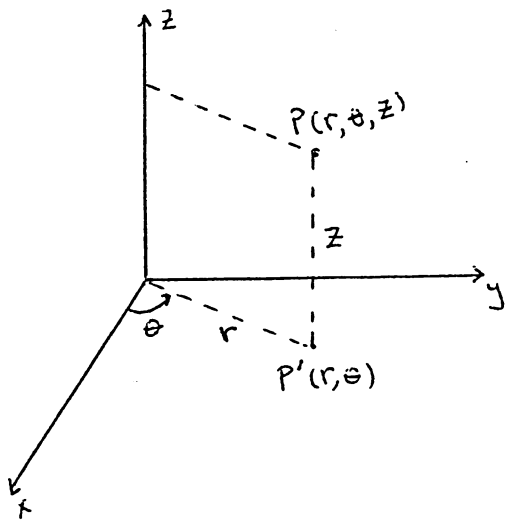
In a left-handed system the z-axis would point in the opposite direction. It is not entirely uncommon to see the z-axis point into the screen in some graphics books. This is often the case when you wish z to represent the concept of depth.

Also, a convention has to be established about what is a positive angle when considering rotation.

The convention we shall use is to look down the axis you're rotating about, toward the origin. Rotation <u>counter-clockwise</u> is considered positive.



CYLINDRICAL COORDINATE SYSTEM



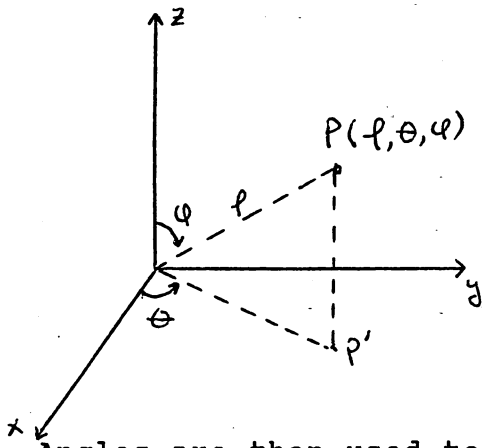
The cylindrical system is basically polar coordinates with the z component added for the 3rd dimension of height.(The point P' is a point in the x-y plane.) The name "cylindrical coordinate system" comes from the fact it is very easy to express the equation of a cylinder in this system.

Consider the set { (r,θ ,z): r=2 }

Imagine a point P in that set, where θ can be any angle, z can be any height, but the point always has to be 2 units away from

the z-axis. If you visualize an "infinitely" long cylindrical tube centered around the z-axis, you've got it!! This undoubtedly is the basis of the name of this system.

SPHERICAL COORDINATE SYSTEM



In actuality, the spherical coordinate system is more (in spirit) like the 2-D polar coordinate system. As you can see from the diagram of the system, we measure the distance to the point directly from the origin. Angles are then used to describe the orientation of that line segment. Since we're in three dimensions we have an extra degree of freedom, so it takes two angles to fix its location. By convention, Φ is an angle such that $0 \leq \Phi \leq \pi$ and $\rho \geq 0$.

The name "spherical" comes from the fact that a sphere can easily be described by $\{ (\rho, \theta, \Phi) : \rho = 10 \}$ for example.

Conversion from cylindrical or spherical coordinates to rectangular coordinates is often a necessity. (It may be theoretically more elegant to use cylindrical or spherical coordinates to describe a situation, but if it has to be adapted to a computer system that "knows" only rectangular coordinates, then conversion is necessary.) Only elementary trig relationships are necessary to make the conversions. The following list come easily by considering the different right triangles involved.

Cylindrical	Spherical
$(r,\theta,z) \rightarrow (x,y,z)$	$(\rho,\theta,\Phi) \rightarrow (x,y,z)$
$x = r\cos\theta$	$x = \rho\sin\Phi\cos\theta$
$y = r\sin\theta$	$y = \rho\sin\Phi\sin\theta$
$z = z$	$z = \rho\cos\Phi$

problems

1.

(a) Establish the formula for calculating distance of a point from the origin in rectangular coordinates.

(b) Find the distance from the origin of P:(2,4,10)

(c) What is the distance between P:(2,4,10) and Q:(-2,7,9)?

2. Draw a couple of other orientations of right handed coordinate systems other than what is shown in the text.

3. Find the rectangular coordinates of the following points:

(a) $(5,\pi/6,10)$ in cylindrical coordinates.

(b) $(7,7\pi/4,-2)$ in cylindrical coordinates.

(c) $(7,\pi/6,\pi/4)$ in spherical coordinates.

(d) $(10,5\pi/6,\pi/3)$ in spherical coordinates.

4. Describe in words the figures these equations describe. Try to sketch a picture of each.

(a) $\{ (r,\theta,z): z=5 \}$ (b) $\{ (\rho,\theta,\Phi): \Phi = \pi/4 \}$

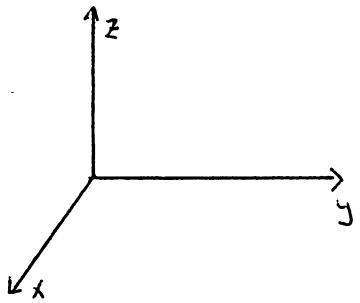
(c) $\{ (\rho,\theta,\Phi): \rho\cos\Phi = 4 \}$ (d) $\{ (r,\theta,z): \theta = \pi/4 \}$

chapter twelve

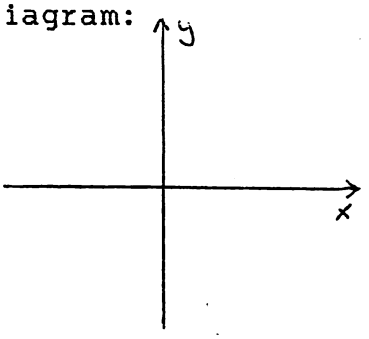
DISPLAYING 3-D ON A 2-D SCREEN

I'll gradually work into this topic. I'll make some simplifing assumptions first, look at a "easy" way to represent 3-D objects then work up to a more realistic representation.

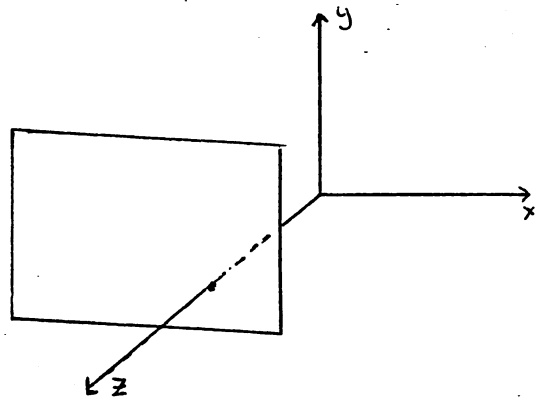
Now, normally, when coordinates (x,y,z) are given(especially in a math book) it is with respect to the following diagram:



However, our screen is usually thought of as:



Whereas the z-coordinate is often used to represent vertical height of a point above the xy plane, we would normally use y to represent height in a two dimensional situation. So, to simplfy things in my programs I will think of 3-D as in the diagram below.

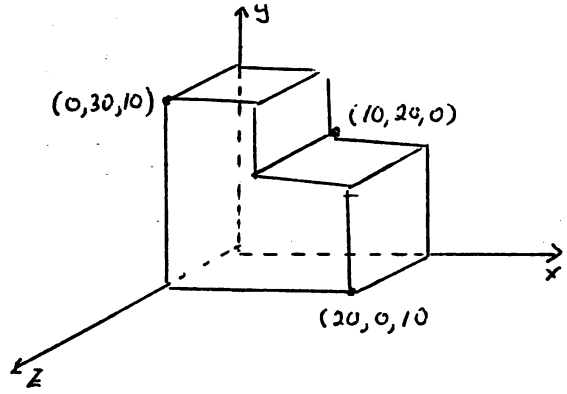


I will think of the TV screen being positioned on the positive z-axis and the viewer being further out on the z-axis. The TV screen will be perpendicular to the z-axis and parallel to the xy plane.

To further simplify things it will be assumed the z-axis will penetrate the sreen in the center.

The viewport will be the entire screen and the window is basically the same as the viewport, but with the origin translated to the center (ie. window = [-xl/2,xl/2]x[-yl/2,yl/2] viewport = [0,xl]x[0,yl]). The coordinates of the 3-D objects will be chosen so that clipping will not normally be a problem.

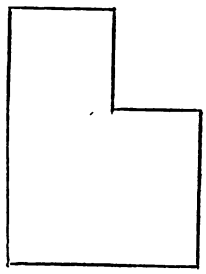
Let's write programs to display a "solid" stairstep object:



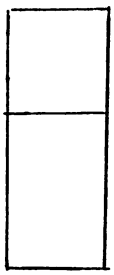
A typical vertex of the object will have coordinates (x,y,z), the "depth being given last.

PARALLEL ORTHOGRAPHIC PROJECTION

The simplest type of representation of a 3-D object is the parallel orthographic projection. (Say that 5 times fast!!!) This type of projection simply ignores the depth information. (ie. (20,0,10) becomes (20,0); (20,0,0) becomes (20,0) also; etc.) The stairstep would look like the diagram to the right (looking down the z-axis).



If we put our TV screen on the x-axis(and made the x coordinate represent depth, we would get the following view: (the x coordinate would be ignored in graphing the point, and only the y and z information would be relevant)



These are views you would see on a blueprint in certain drafting applications. Given several "side" views one can learn to interpret what the 3-D object must look like.

See the program at the end of this chapter demonstrating parallel orthographic projections. It will give you a "front" (looking down the z-axis) view, a "side" (looking down the x-axis) view, and a "top" (looking down the y-axis) view of the object.

You could add more objects at the end of the program. Many variations to this program could be added for your own purposes and pleasure.

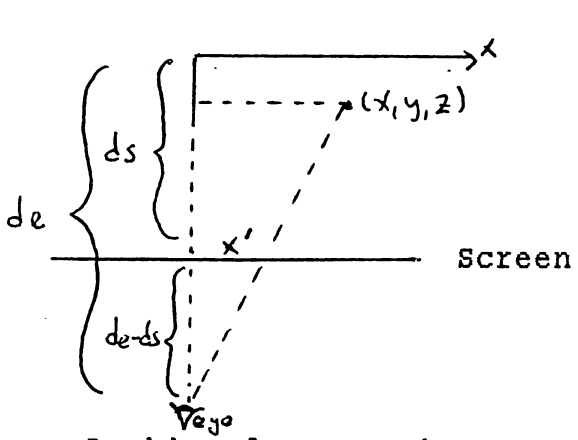
The subroutines used are fairly obvious. The subroutine called Position_object places the object on the screen. The H and K allow you to position various views at different places on the screen. It is a combination windowing function and translation routine. The subroutine called change_views "renames" the coordinate variables for the various views.(Remember, everything is "x and y" to the computer screen.)

Can you determine what object1 is before you run the second program? You may find it hard to determine an objects shape from these three parallel projections, but I'm told that with practice it's possible!

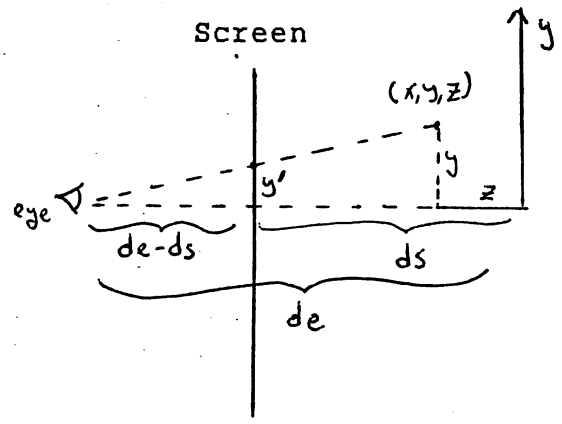
PERSPECTIVE PROJECTION

It's pretty obvious that parallel orthographic projections are not what most people consider as a 3-D representation of an object, however. Generally one usually wants to include perspective in representing an object to give an illusion of realistic depth. The idea of perspective is that as one moves an object further away from the viewer, it will appear to be smaller. Thus the back face of a cube will appear smaller than the front face, railroad tracks will appear to be closer together as one views them receeding, etc. There are different types of perspective schemes, but I won't bother to identify all the different types in this book.

To see how a point projects on the screen(under our simplifing assumptions made earlier) consider the following diagrams: (diagrams made by "god" observing the object, the axis, and the observer)



Looking down y-axis



Looking down x-axis

Considering similar triangles in the above pictures:

$$\frac{x'}{de-ds} = \frac{x}{de-z} \qquad x' = \frac{x(de-ds)}{de-z} \qquad \frac{y'}{de-ds} = \frac{y}{de-z} \qquad y' = \frac{y(de-ds)}{de-z}$$

As you can see from the diagram, the distance to the eye(De) and the distance to the screen(Ds) are measured from the origin of our three dimensional space(from which our object is referenced). This is not always the most natural from the viewer's point of view, but in the interest of the famous kiss principle I'll not try to interact a viewer's coordinate system with the object's coordinate system.

Now examine the second program at the end of this chapter. I tried to keep it almost the same as the previous program. The subroutine perspective_window contains the perspective formulas developed previously and incorporates the windowing function. The subroutine scale_translate_object scales then translates the object before we view it.

You have to play with the values for de and ds to get the perspective you want. Also be careful of the scale factor or

you'll find yourself inside the object. Also you might find some points being off screen. This won't really be a problem for this computer, as GEM usually will "plot points off screen" with no error messages. This is not true universally!! If you modified this program to work within a smaller viewport other than the whole screen, then you would want to think about incorporating the clipping subroutine.

Since both of my objects built into the program are "tied" to the origin, I've applied the scaling routine first, then translation. I think that will seem to give the desired results most naturally.

There are no hidden lines, so sometimes the lines in "back" confuse the eye. Sometimes you really have to blink the eyes to make it look the way you know it has to look!

Feel free to modify the programs, put in your own objects. Other books may help you to jazz up the programs so that hidden lines and faces can be determined and not drawn. The hidden line-face problem is not a simple one, and I've decided not to tackle it in this introductory book.

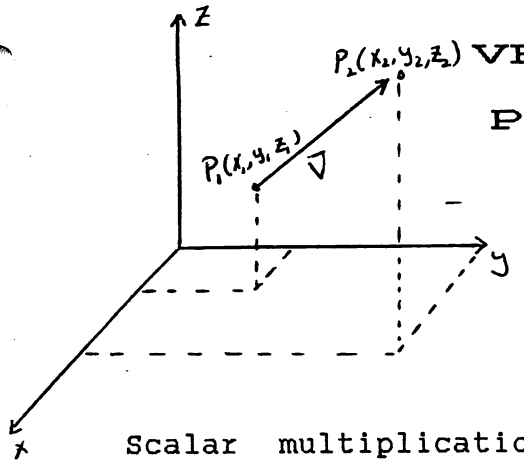
```
' A simple demo of three-dimensional objects
'      Parallel Orthographic Projection
object0:
DATA 0,0,10,20,0,10,20,20,10,10,20,10,10,30,10,0,30,10,0,0,10,-1111,0,0
DATA 0,0,0,20,0,0,20,20,0,10,20,0,10,30,0,0,30,0,0,0,0,-1111,0,0
DATA 20,0,10,20,0,0,-1111,0,0
DATA 10,20,10,10,20,0,-1111,0,0,10,30,10,10,30,0,-1111,0,0
DATA 0,30,10,0,30,0,-1111,0,0,0,0,10,0,0,0,-9999,0,0
object1:
DATA 0,0,0,0,0,50,50,0,50,50,0,0,0,0,0,25,25,25,0,0,50,-1111,0,0
DATA 50,0,0,25,25,25,50,0,50
DATA -9999,0,0
'
LET xc=320
LET yc=80  ! Viewport  [0,2*xc]x[0,2*yc]
LET b=200
LET d=100   ! window  [-b,b]x[-d,d]
'
DO
  CLS
  draw_axis
  set_up_textwindow
  INPUT "enter object # (0 or 1) to graph(-99 to end) >>",object
 EXIT IF object=-99
  '
  set_up_textwindow
  PRINT "The front view, no translation, no scaling"
  COLOR 1
  plot_object(1,1,1,0,0,"F")
  firsttime!=TRUE
  PAUSE 200
  DO
    set_up_textwindow
    INPUT "Enter Scale factor--Sx,Sy,Sz >>",sx,sy,sz
    INPUT "Enter Translation--horizontal,vertical >>",h,k
    INPUT "Enter view--Front,Side,or Top(F,S,or T) >>",view$
    IF firsttime! THEN
      COLOR 0  ! Erase original figure
      plot_object(1,1,1,0,0,"F")
      draw_axis
      firsttime!=FALSE
    ENDIF
    COLOR 1
    plot_object(sx,sy,sz,h,k,UPPER$(view$))
    set_up_textwindow
    PRINT "CHANGE OBJECTS OR CLEAR SCREEN?(y,or n) ";
    answer$=INPUT$(1)
    EXIT IF UPPER$(answer$)="Y"
  LOOP
LOOP
PROCEDURE draw_axis
  COLOR 1
  DRAW 0,yc TO 2*xc,yc  ! x axis
  DRAW xc,0 TO xc,2*yc  ! y axis
  PRINT AT(77,11);b
  PRINT AT(40,1);d
RETURN
```

```
PROCEDURE restore_data
  IF object=0 THEN
    RESTORE object0
  ELSE
    RESTORE object1
  ENDIF
RETURN
PROCEDURE set_up_textwindow
  FOR i=21 TO 25
    PRINT AT(1,i);SPACE$(79);
  NEXT i
  PRINT AT(1,21);
RETURN
PROCEDURE plot_object(xscale,yscale,zscale,horiz,vert,view$)
  restore_data
  DO
    READ x,y,z
    scale_object(xscale,yscale,zscale)
    change_views(view$)
    position_object(horiz,vert)
    PLOT x,y
    DO
      READ x,y,z
      EXIT IF x=-9999 OR x=-1111
      scale_object(xscale,yscale,zscale)
      change_views(view$)
      position_object(horiz,vert)
      DRAW  TO x,y
    LOOP
    EXIT IF x=-9999
  LOOP
RETURN
PROCEDURE scale_object(xscale,yscale,zscale)
  LET x=x*xscale
  LET y=y*yscale
  LET z=z*zscale
RETURN
PROCEDURE change_views(view$)
  IF view$="S" THEN
    LET x=z
  ENDIF
  IF view$="T" THEN
    LET y=z
  ENDIF
RETURN
PROCEDURE position_object(horiz,vert)
  LET x=x+horiz  ! Apply the translation
  LET y=y+vert
  LET x=(x+b)/b*xc ! Windowing function
  LET y=(d-y)/d*yc
RETURN
```

```
' A simple demo of three-dimensional objects
'      With Perspective
object0:
DATA 0,0,10,20,0,10,20,20,10,10,20,10,10,30,10,0,30,10,0,0,10,-1111,0,0
DATA 0,0,0,20,0,0,20,20,0,10,20,0,10,30,0,0,30,0,0,0,0,-1111,0,0
DATA 20,0,10,20,0,0,-1111,0,0
DATA 10,20,10,10,20,0,-1111,0,0,10,30,10,10,30,0,-1111,0,0
DATA 0,30,10,0,30,0,-1111,0,0,0,0,10,0,0,0,-9999,0,0
object1:
DATA 0,0,0,0,0,50,50,0,50,50,0,0,0,0,0,25,25,25,0,0,50,-1111,0,0
DATA 50,0,0,25,25,25,50,0,50
DATA -9999,0,0
'
LET xc=320
LET yc=80  ! Viewport  [0,2*Xc]x[0,2*Yc]
LET b=200
LET d=100   ! window  [-B,B]x[-D,D]
'
DO
  CLS
  draw_axis
  set_up_textwindow
  INPUT "enter object # (0 or 1) to graph(-99 to end) >>",object
  EXIT IF object=-99
  '
  set_up_textwindow
  LET de=300
  LET ds=100
  PRINT "Scale of 1, no translation, distance to eye is ";de
  PRINT "Distance to screen is ";ds
  COLOR 1
  plot_object(1,0,0,0)
  firsttime!=TRUE
  PAUSE 200
  DO
    set_up_textwindow
    INPUT "Enter overall Scale factor--Sf >>",sf
    INPUT "Enter Translation in X and Y and Z direction >>",h,k,l
    PRINT "Present distance to eye, distance to screen  is ";de,ds
    INPUT "Enter distance to eye,distance to screen >> ";de,ds
    IF firsttime! THEN
      COLOR 0  ! Erase original figure
      plot_object(1,0,0,0)
      draw_axis
      firsttime!=FALSE
    ENDIF
    COLOR 1
    plot_object(sf,h,k,l)
    set_up_textwindow
    PRINT "CHANGE OBJECTS OR CLEAR SCREEN?(y,or n) ";
    answer$=INPUT$(1)
    EXIT IF UPPER$(answer$)="Y"
  LOOP
LOOP
PROCEDURE draw_axis
  COLOR 1
  DRAW 0,yc TO 2*xc,yc  ! x axis
  DRAW xc,0 TO xc,2*yc  ! y axis
```

```
    PRINT AT(77,11);b
    PRINT AT(40,1);d
  RETURN
  PROCEDURE restore_data
    IF object=0 THEN
      RESTORE object0
    ELSE
      RESTORE object1
    ENDIF
  RETURN
  PROCEDURE set_up_textwindow
    FOR i=21 TO 25
      PRINT AT(1,i);SPACE$(79);
    NEXT i
    PRINT AT(1,21);
  RETURN
  PROCEDURE plot_object(scale,h,k,l)
    restore_data
    DO
      READ x,y,z
      scale_translate_object(scale,h,k,l)
      perspective_window
      IF NOT inside_figure! THEN
        PLOT x,y
      ENDIF
      DO
        READ x,y,z
        EXIT IF x=-9999 OR x=-1111
        scale_translate_object(scale,h,k,l)
        perspective_window
        IF NOT inside_figure! THEN
          DRAW  TO x,y
        ENDIF
      LOOP
  EXIT IF x=-9999
LOOP
  RETURN
  PROCEDURE scale_translate_object(scale,h,k,l)
    LET x=x*scale+h
    LET y=y*scale+k
    LET z=z*scale+l
  RETURN
  PROCEDURE perspective_window
    LET inside_figure!=FALSE
    LET d1=de-z
    IF d1>0 THEN
      LET x=x*(de-ds)/d1
      LET y=y*(de-ds)/d1
    ELSE
      LET inside_figure!=TRUE
    ENDIF
    LET x=(x+b)/b*xc ! Windowing function
    LET y=(d-y)/d*yc
  RETURN
```

chapter thirteen



VECTORS, LINES AND PLANES IN 3D

A vector $\bar{v}$ from P_1 to P_2 is denoted by $\bar{v} = \langle x_2-x_1, y_2-y_1, z_2-z_1 \rangle$. x_2-x_1, y_2-y_1, z_2-z_1 are called the components of $\bar{v}$.

Scalar multiplication and vector addition is defined as it was for the two dimensional vectors.

EQUATION OF A LINE

As in the two dimensional case, the parametric equations

$$x = x_1 + (x_2-x_1)t$$

$$y = y_1 + (y_2-y_1)t$$

$$z = z_1 + (z_2-z_1)t$$

represent a line going through points P_1 and P_2. If $0<t<1$, then we have the line segment between P_1 and P_2.

It is not uncommon to define the concept of "adding" a point and a vector to get another point. If you look at the diagram above, you can see that starting with point P_1, adding to it the components of $\bar{v}$, the coordinates of P_2 emerge.

(ie. $x_1 + (x_2 - x_1) = x_2$, $y_1 + (y_2 - y_1) = y_2$, $z_1 + (z_2 - z_1) = z_2$)

So, if $P=(a,b,c)$ is a point and $\bar{v}$ is a vector with components $\langle v_1, v_2, v_3 \rangle$, then $P + \bar{v} = (a+v_1, b+v_2, c+v_3)$.

Now, the parametric equations can be written as $(x,y,z) = P + t\bar{v}$ where (x,y,z) is any point on the line. (By the way, this could have been done in the two dimensional case also. I just chose not to throw everything in all at one time!!)

problems

1. What are the parametric equations of a line that contains the points (2,3,-4) and (-2,5,7)?

2. What are the parametric equations of the line segment between (3,-4,6) and (4,3,-7)?

3. A point (3,4,-2) lies on line L. A vector <-2,3,5> is parallel to the line. Find the parametric equations for L.

DOT PRODUCT

Let $\bar{v} = \langle v_1, v_2, v_3 \rangle$ and $\bar{u} = \langle u_1, u_2, u_3 \rangle$ be two vectors. The **dot product** $\bar{u} \cdot \bar{v}$ is a <u>number</u> given by:

$$\bar{u} \cdot \bar{v} = u_1v_1 + u_2v_2 + u_3v_3.$$

(The dot product could have been defined similarly for a two dimensional vector.)

By the Pythagorean theorem, it is easy to see that the **length** of a vector $\bar{v}=\langle v_1, v_2, v_3 \rangle$ is given by $\sqrt{v_1^2 + v_2^2 + v_3^2}$. $|\bar{v}|$ is often the symbol used for the length of $\bar{v}$. ($|\bar{v}|$ is sometimes called the norm of the vector $\bar{v}$.)

Dot products are fun, and there are some easy properties to prove. Can you prove these?

(a) $\bar{v} \cdot \bar{v} = |\bar{v}|^2$

(b) $\bar{u} \cdot \bar{v} = \bar{v} \cdot \bar{u}$ (commutative property)

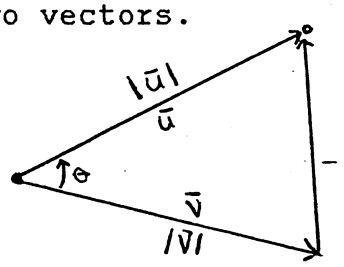
(c) $\bar{u} \cdot (\bar{v}+\bar{w}) = \bar{u} \cdot \bar{v} + \bar{u} \cdot \bar{w}$ (distributive property)

(d) $(t\bar{u}) \cdot \bar{v} = t(\bar{u} \cdot \bar{v}) = \bar{u} \cdot (t\bar{v})$

(e) $|\bar{u} \cdot \bar{v}| \leq |\bar{u}||\bar{v}|$

(f) $\frac{1}{|\bar{v}|}\bar{v}$ has length 1. (a unit vector)

The dot product can be useful in expressing the angle between two vectors.



By the law of cosines,

$$|\bar{u}-\bar{v}|^2 = |\bar{u}|^2+|\bar{v}|^2 - 2|\bar{u}||\bar{v}|\cos\theta$$

Now, simplifying the above equation using the rules about dot products, you get these equivalent equations:

$$(\bar{u}-\bar{v})\cdot(\bar{u}-\bar{v}) = \bar{u}\cdot\bar{u} + \bar{v}\cdot\bar{v} -2|\bar{u}||\bar{v}|\cos\theta$$

$$\bar{u}\cdot\bar{u} -2\bar{u}\cdot\bar{v} + \bar{v}\cdot\bar{v} = \bar{u}\cdot\bar{u} + \bar{v}\cdot\bar{v} -2|\bar{u}||\bar{v}|\cos\theta$$

$$-2\bar{u}\cdot\bar{v} = -2|\bar{u}||\bar{v}|\cos\theta$$

$$\boxed{\cos\theta = \frac{\bar{u}\cdot\bar{v}}{|\bar{u}||\bar{v}|}}, \quad 0\le\theta\le180^\circ$$

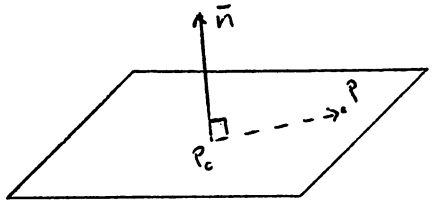
Since $\cos 90^\circ = 0$, we can formulate this rule: $\bar{u}$ and $\bar{v}$ are perpendicular whenever $\bar{u}\cdot\bar{v} = 0$. We'll have an occasion to use this later on!

problems

1. Let $\bar{v} = \langle 3,4,7\rangle$ and $\bar{u} = \langle 4,5,-4\rangle$. find $\bar{u}\cdot\bar{v}$ and $|\bar{u}|$ and $|\bar{v}|$.
2. Let $\bar{v} = \langle 4,8,-2\rangle$. Find the length of $\bar{v}$.
3. Let $\bar{u} = \langle 5,4,2\rangle$. Find $\bar{u}\cdot\bar{u}$. Find a unit vector in the direction of $\bar{u}$.
4. What is the cosine of the angle between $\langle 2,-4,5\rangle$and $\langle -3\ 2,-7\rangle$?
5. Find the angle between $\langle 3,2,4\rangle$ and $\langle 2,5,-4\rangle$.
6. Let $\bar{u} = \langle 3,5,-8\rangle$. Find a nonzero vector that is perpendicular to $\bar{u}$.

EQUATION OF A PLANE

To fix a particular plane in space, all we need to know is a point $P_o = (x_o, y_o, z_o)$ on the plane and a vector $\bar{n} = \langle a,b,c \rangle$ that is perpendicular to the plane. (There are many planes through a point, but the vector n will fix it in only one direction.)



Let P(x,y,z) be any point on the plane. The vector $\bar{n}$ will be perpendicular to any vector on the plane. In particular $\bar{n}$ is perpendicular to the vector from P_o to P_1.

So, $\bar{n} \cdot \langle x-x_o, y-y_o, z-z_o \rangle = 0$ or $\langle a,b,c \rangle \cdot \langle x-x_o, y-y_o, z-z_o \rangle = 0$

Finally, $a(x-x_o)+b(y-y_o)+c(z-z_o)=0$

This is considered an equation of a plane, containing the point P_o and with a vector n perpendicular to the plane. (By the way, sometimes we say n is normal to the plane, rather than saying it is perpendicular.)

problems

1. Find the equation of a plane containing (0,0,0) and is normal to $\langle 0,0,1 \rangle$.
2. Find the equation of the plane containing (2,-4,3) is is perpendicular to $\langle -1,2,7 \rangle$.
3. Find the equation of the plane containing (3,5,7) and is parallel to the plane given by $2x-5y+3z=4$.
4. Find the equation of the plane containing (2,1,3), (-1,2,1), and (1,-2,2). (This is harder than the previous 3 problems!)

CROSS PRODUCT OF TWO VECTORS

Knowing how to find a vector that is perpendicular to two given vectors $\bar{u}$ and $\bar{v}$ is a handy thing when working in 3-D geometry. The vector we will find that is perpendicular will be called the cross product (denoted by $\bar{u} \times \bar{v}$) of the vectors involved. Let's develop a way to find a perpendicular vector.

Let $\bar{u} = \langle u_1, u_2, u_3 \rangle$ and $\bar{v} = \langle v_1, v_2, v_3 \rangle$ be two vectors. Suppose the vector $\bar{n} = \langle a, b, c \rangle$ is to be perpendicular to $\bar{u}$ and $\bar{v}$. For this to be true we would have $\bar{n}\ \bar{u}=0$ and $\bar{n}\ \bar{v}=0$.

$$au_1 + bu_2 + cu_3 = 0$$

and

$$av_1 + bv_2 + cv_3 = 0$$

or equivalently $au_1 + bu_2 = -cu_3$

$$av_1 + bv_2 = -cv_3$$

Solve these equations simultaneously for a and b:

$$\begin{array}{ll} \text{(mult. by } v_1\text{)} & au_1v_1 + bu_2v_1 = -cu_3v_1 \\ \text{(mult. by } -u_1\text{)} & -au_1v_1 - bu_1v_2 = cu_1v_3 \\ \hline & b(u_2v_1 - u_1v_2) = c(u_1v_3 - u_3v1) \end{array}$$

$$b = \frac{u_1v_3 - u_3v_1}{u_2v_1 - u_1v_2}\,c \quad = \frac{u_3v_1 - u_1v_3}{u_1v_2 - u_2v_1}\,c$$

Similarly, solve for a.

$$\begin{array}{ll} \text{(mult. by } v_2\text{)} & au_1v_2 + bu_2v_2 = -cu_3v_2 \\ \text{(mult. by } -u_2\text{)} & -au_2v_1 - bu_2v_2 = cu_2v_3 \\ \hline & a(u_1v2 - u_2v_1) = c(u_2v_3 - u_3v_2) \end{array}$$

$$a = \frac{u_2v_3 - u_3v_2}{u_1v2 - u_2v_1}\,c$$

There are infinitely many vectors perpendicular to a given pair of vectors(could have various lengths, be 180 degrees from each other). This wide variety of different vectors is expressed in the above equations in that there are infinitely many values for a and b, depending on what you choose for the value of c. It turns

out that a particularly good choice for c is to let c be the value of the denominators in the above fractions. Thus let $c = u_1v_2 - u_2v_1$. For this choice of c, $a = u_2v_3 - u_3v_2$ and $b = u_3v_1 - u_1v_3$. This vector is called the cross product.

Now I know you're thinking: "There's no way in the world anyone can remember that!!" Aha--that's what you think!! Enter the world of determinants. I'll take you on a short detour, then come back to the cross product.

DETERMINANTS

A 2x2 determinant is four numbers arranged in the following pattern:

$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$ It is a number given by

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad-bc$$

A 3x3 determinant (and higher order determinants) are found in the following manner:

$$\begin{vmatrix} a & b & c \\ u & v & w \\ x & y & z \end{vmatrix} = a\begin{vmatrix} v & w \\ y & z \end{vmatrix} - b\begin{vmatrix} u & w \\ x & z \end{vmatrix} + c\begin{vmatrix} u & v \\ x & y \end{vmatrix}$$

$$\begin{vmatrix} \textcircled{a} & -b & -c \\ u & v & w \\ x & y & z \end{vmatrix} \quad \begin{vmatrix} a & \textcircled{-b} & -c \\ u & v & w \\ x & y & z \end{vmatrix} \quad \begin{vmatrix} a & -b & \textcircled{-c} \\ u & v & w \\ x & y & z \end{vmatrix}$$

For the higher order determinants, follow the same pattern and continue to alternate signs as you go across the top row.

Determinants have many interesting properties, and I have only given you the bare necessities to be able to calculate the value of a determinant. But all the interesting stuff is not necessary for our immediate purposes. Now let's go back to the subject of cross products.

RETURN TO CROSS PRODUCTS

Now recall, $\bar{u} = \langle u_1, u_2, u_3 \rangle$

$\bar{v} = \langle v_1, v_2, v_3 \rangle$

We found $a = u_2v_3 - u_3v_2$ $\quad b = u_3v_1 - u_1v_3$ $\quad c = u_1v_2 - u_2v_1$.

Using determinants: $a = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix}$ $\quad b = -\begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix}$ $\quad c = \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}$

There is a nice "device" to get these 2x2 determinants by using the way we figure out 3x3 determinants. Toward this purpose, let me introduce a new notation for vectors:

Note that any vector $\langle p,q,r \rangle$ can be written as $p\langle 1,0,0 \rangle + q\langle 0,1,0 \rangle + q\langle 0,0,1 \rangle$. The vectors $\langle 1,0,0 \rangle$ and $\langle 0,1,0 \rangle$ and $\langle 0,0,1 \rangle$ are unit vectors pointing along the x-axis, y-axis, and z-axis respectively. These vectors have been given special names:

$\bar{i} = \langle 1,0,0 \rangle$ $\quad \bar{j} = \langle 0,1,0 \rangle$ $\quad \bar{k} = \langle 0,0,1 \rangle$.

So our vector $\langle p,q,r \rangle = p\bar{i} + q\bar{j} + r\bar{k}$ in our new notation.

(Example: $\langle 3,4,5 \rangle = 3\bar{i}+2\bar{j}+4\bar{k}$; $\quad \langle -2,1,3 \rangle = -2\bar{i}+\bar{j}+3\bar{k}$; etc.)

Ok, it now turns out that $\bar{u} \times \bar{v} = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix}$

To calculate, go across the top row as in our example in evaluating a 3x3 determinant.

$$\bar{u} \times \bar{v} = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix}\bar{i} - \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix}\bar{j} + \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}\bar{k}$$ Slick !!!!

EXAMPLE: Let $\bar{u} = \langle 2,1,3 \rangle$ and $\bar{v} = \langle 1,4,-2 \rangle$

$$\bar{u} \times \bar{v} = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ 2 & 1 & 3 \\ 1 & 4 & -2 \end{vmatrix} = \begin{vmatrix} 1 & 3 \\ 4 & -2 \end{vmatrix}\bar{i} - \begin{vmatrix} 2 & 3 \\ 1 & -2 \end{vmatrix}\bar{j} + \begin{vmatrix} 2 & 1 \\ 1 & 4 \end{vmatrix}\bar{k}$$

$$= (-2-12)\bar{i} - (-4-3)\bar{j} + (8-1)\bar{k}$$

$$= -14\bar{i} + 7\bar{j} + 7\bar{k} = \langle -14,7,7 \rangle$$

PROPERTIES OF CROSS PRODUCTS

Let me end by summarizing some properties of cross products. Some might be kind of hard to prove, but all should be fun to try to prove.

(a) $\bar{u} \times \bar{v} = -(\bar{v} \times \bar{u})$ (ie. $\bar{u} \times \bar{v}$ is a vector in opposite direction of $\bar{v} \times \bar{u}$

(b) $\bar{u} (\bar{v} \times \bar{w}) = (\bar{u} \times \bar{v}) \bar{w}$ (note: there is no "normal" associative property of just cross products)

(c) $\bar{u} \times (\bar{v}+\bar{w}) = (\bar{u} \times \bar{v})+(\bar{u} \times \bar{w})$ (ie. distributive prop.)

$(\bar{u}+\bar{v}) \times \bar{w} = (\bar{u} \times \bar{w})+(\bar{v} \times \bar{w})$

(d) $(t\bar{u}) \times \bar{v} = t(\bar{u} \times \bar{v}) = \bar{u} \times (t\bar{v})$

(e) $|\bar{u} \times \bar{v}| = |\bar{u}||\bar{v}|\sin\theta$

(f) $\bar{u}$ and $\bar{v}$ are parallel iff $\bar{u} \times \bar{v} = \bar{0}$

You now have most of the tools to tackle explanations and problems in 3-D space. Good luck!

problems

1. Evaluate these determinants:

(a) $\begin{vmatrix} 3 & 2 \\ 1 & 4 \end{vmatrix}$ (b) $\begin{vmatrix} 1 & -2 \\ 4 & 7 \end{vmatrix}$ (c) $\begin{vmatrix} 3 & 2 & 1 \\ 1 & 4 & 1 \\ 2 & 1 & 2 \end{vmatrix}$ (d) $\begin{vmatrix} 1 & 2 & 1 \\ 4 & 3 & 2 \\ 2 & 1 & 1 \end{vmatrix}$

2. Exchange the 2nd and 3rd row in (d) above. Recalculate new determinant. Is the result "hinted at" in the properties above? Explain.

3. Find $\bar{u} \times \bar{v}$ for these vectors:

(a) $\bar{u}=\langle 1,2,3\rangle$ $\bar{v}=\langle 1,-3,2\rangle$ (b) $\bar{u}=\langle 1,-1,2\rangle$ $\bar{v}=\langle -2,1,3\rangle$

(c) $\bar{u}=\langle 7,2,1\rangle$ $\bar{v}=\langle -2,3,-2\rangle$

4. Find $\bar{i} \times \bar{j}$; $\bar{j} \times \bar{k}$; $\bar{i} \times \bar{k}$

chapter fourteen

3-D TRANSFORMATIONS

This follows along almost exactly the same as our previous two dimensional discussion. I'll only elaborate on any major differences.

TRANSLATION

To include translation as a matrix topic, we must go to homogeneous coordinates: $(x,y,z)=(x,y,z,1)$

$$[x\ y\ z\ 1]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ H & K & L & 1\end{bmatrix} = [x+H\ \ y+K\ \ z+L\ \ 1]$$

SCALING

$$[x\ y\ z]\begin{bmatrix}S_x & 0 & 0\\ 0 & S_y & 0\\ 0 & 0 & S_z\end{bmatrix} = [S_xx\ \ S_yy\ \ S_zz]$$

REFLECTION

About x-y plane:

$$[x\ y\ z]\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & -1\end{bmatrix} = [x\ y\ -z]$$

About x-z plane:

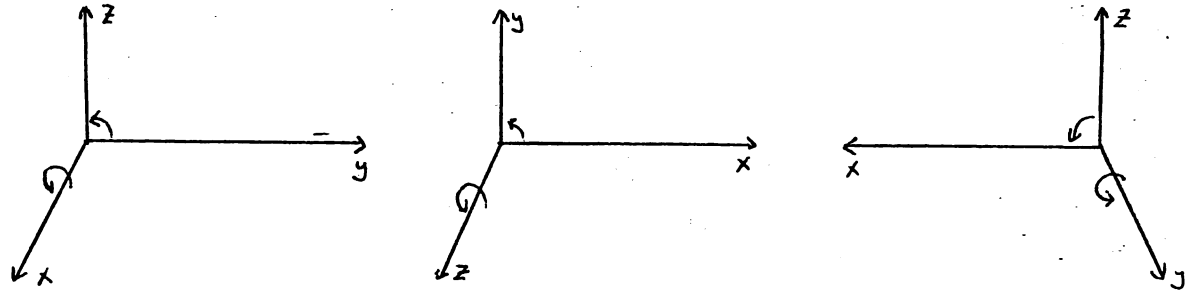
$$[x\ y\ z]\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix} = [x\ -y\ z]$$

About y-z plane:

$$[x\ y\ z]\begin{bmatrix}-1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} = [-x\ y\ z]$$

ROTATION

We have 3 types of rotation--rotation about each of the axis.



Above are pictures of the axis in 3 different views, each positioned so you can imagine looking down the x, z, and y axis respectively. Recall that positive rotation is counterclockwise from this view. Obviously(?) rotation around a particular axis doesn't affect that coordinate. (ie. rotation around the z-axis doesn't affect the z-coordinate of a point, etc.)

The first two pictures above look exactly like the diagram we worked from to derive the rotation formula in two dimension. Thus we get:

ROTATION ABOUT X-AXIS

$$[x'\ y'\ z'] = [x\ y\ z] \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

$$\begin{aligned} x' &= x \\ y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \end{aligned}$$

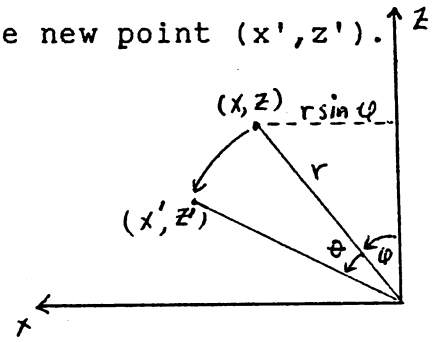
ROTATION ABOUT Z-AXIS

$$[x'\ y'\ z'] = [x\ y\ z] \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \\ z' &= z \end{aligned}$$

ROTATION ABOUT Y-AXIS

This picture looks different!! We do get slightly different results. Suppose we start with a point (x,z) at some angle Φ, as shown in the picture below. Now rotate it by an angle θ to get the new point (x',z').



x=rsinΦ and z=rcosΦ

x =rsin(Φ+θ) and z =rcos(Φ+θ)

Simplifying x and z we get:

$$x' = r(\sin\Phi\cos\theta + \cos\Phi\sin\theta) = r\sin\Phi\cos\theta + r\cos\Phi\sin\theta$$
$$= x\cos\theta + z\sin\theta$$

$$z' = r(\cos\Phi\cos\theta - \sin\Phi\sin\theta) = r\cos\Phi\cos\theta - r\sin\Phi\sin\theta$$
$$= z\cos\theta - x\sin\theta$$
$$= -x\sin\theta + z\cos\theta$$

So

$$[x'\ y'\ z'] = [x\ y\ z]\begin{bmatrix}\cos\theta & 0 & -\sin\theta\\ 0 & 1 & 0\\ \sin\theta & 0 & \cos\theta\end{bmatrix}$$

$$x' = x\cos\theta + z\sin\theta$$
$$y' = y$$
$$z' = -x\sin\theta + z\cos\theta$$

Ok, let's add these formulas for rotation to our previous ThreeD program in chapter 12. Rotation will be asked for in the input part of the program.

The program is unsophisticated in that you can not change the order of the way the transformations are applied--and order does make a difference!! The program applies the transformations in the order that it asks for in the input!

As always, feel free to improve this program. As with most learning, you learn best by doing!

```
' A simple demo of three-dimensional objects
'      With Perspective and Rotation added
object0:
DATA 0,0,10,20,0,10,20,20,10,10,20,10,10,30,10,0,30,10,0,0,10,-1111,0,0
DATA 0,0,0,20,0,0,20,20,0,10,20,0,10,30,0,0,30,0,0,0,0,-1111,0,0
DATA 20,0,10,20,0,0,-1111,0,0
DATA 10,20,10,10,20,0,-1111,0,0,10,30,10,10,30,0,-1111,0,0
DATA 0,30,10,0,30,0,-1111,0,0,0,0,10,0,0,0,-9999,0,0
object1:
DATA 0,0,0,0,0,50,50,0,50,50,0,0,0,0,0,25,25,25,0,0,50,-1111,0,0
DATA 50,0,0,25,25,25,50,0,50
DATA -9999,0,0
'
LET xc=320
LET yc=80  ! Viewport  [0,2*xc]x[0,2*yc]
LET b=200
LET d=100   ! window  [-b,b]x[-d,d]
'
DO
  CLS
  draw_axis
  set_up_textwindow
  INPUT "enter object # (0 or 1) to graph(-99 to end) >>",object
  EXIT IF object=-99
  '
  set_up_textwindow
  LET de=300
  LET ds=100
  PRINT "Scale of 1,no rotation, no translation, distance to eye is ";de
  PRINT "Distance to screen is ";ds
  COLOR 1
  trig_calculations(0,0,0)
  plot_object(1,0,0,0)
  firsttime!=TRUE
  PAUSE 200
  DO
    set_up_textwindow
    INPUT "Enter overall Scale factor--Sf >>",sf
    INPUT "Enter rotation angle about x-axis,y-axis,z-axis >>",theta_x,theta_y,theta_z
    INPUT "Enter Translation in X and Y and Z direction >>",h,k,l
    PRINT "Present distance to eye, distance to screen  is ";de,ds
    INPUT "Enter distance to eye,distance to screen >> ";de,ds
    IF firsttime! THEN
      COLOR 0  ! Erase original figure
      trig_calculations(0,0,0)
      plot_object(1,0,0,0)
      draw_axis
      firsttime!=FALSE
    ENDIF
    COLOR 1
    trig_calculations(theta_x,theta_y,theta_z)
    plot_object(sf,h,k,l)
    set_up_textwindow
    PRINT "CHANGE OBJECTS OR CLEAR SCREEN?(y,or n) ";
    answer$=INPUT$(1)
    EXIT IF UPPER$(answer$)="Y"
  LOOP
LOOP
```

```
PROCEDURE draw_axis
  COLOR 1
  DRAW 0,yc TO 2*xc,yc  ! x axis
  DRAW xc,0 TO xc,2*yc  ! y axis
  PRINT AT(77,11);b
  PRINT AT(40,1);d
RETURN
PROCEDURE restore_data
  IF object=0 THEN
    RESTORE object0
  ELSE
    RESTORE object1
  ENDIF
RETURN
PROCEDURE set_up_textwindow
  FOR i=20 TO 25
    PRINT AT(1,i);SPACE$(79);
  NEXT i
  PRINT AT(1,20);
RETURN
PROCEDURE plot_object(scale,h,k,l)
  restore_data
  DO
    READ x,y,z
    scale_rotate_translate_object(scale,h,k,l)
    perspective_window
    IF NOT inside_figure! THEN
      PLOT x,y
    ENDIF
    DO
      READ x,y,z
      EXIT IF x=-9999 OR x=-1111
      scale_rotate_translate_object(scale,h,k,l)
      perspective_window
      IF NOT inside_figure! THEN
        DRAW  TO x,y
      ENDIF
    LOOP
    EXIT IF x=-9999
  LOOP
RETURN
PROCEDURE scale_rotate_translate_object(scale,h,k,l)
  LET x=x*scale
  LET y=y*scale
  LET z=z*scale
  '  rotation applied in order :about x-axis, then about y-axis
  '                             then about z-axis
  LET yy=y*ctx-z*stx
  LET z=y*stx+z*ctx
  LET y=yy
  LET xx=x*cty+z*sty
  LET z=-x*sty+z*cty
  LET x=xx
  LET xx=x*ctz-y*stz
  LET y=x*stz+y*ctz
  LET x=xx
  ' Now translation
  LET x=x+h
```

```
  LET y=y+k
  LET z=z+l
RETURN
PROCEDURE perspective_window
  LET inside_figure!=FALSE
  LET d1=de-z
  IF d1>0 THEN
    LET x=x*(de-ds)/d1
    LET y=y*(de-ds)/d1
  ELSE
    LET inside_figure!=TRUE
  ENDIF
  LET x=(x+b)/b*xc ! Windowing function
  LET y=(d-y)/d*yc
RETURN
PROCEDURE trig_calculations(ax,ay,az)
  stx=SIN(ax*PI/180)
  ctx=COS(ax*PI/180)
  sty=SIN(ay*PI/180)
  cty=COS(ay*PI/180)
  stz=SIN(az*PI/180)
  ctz=COS(az*PI/180)
RETURN
```

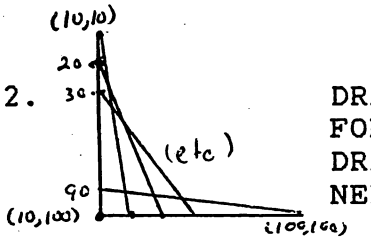
APPENDIX

ANSWERS

PAGE 1-3

1. PLOT 50,i 2. PLOT 110-i,50 3. PLOT i,10
PLOT i,100
PLOT 100,i
PLOT 10,i

PAGE 1-5



2.
```
DRAW 10,10 TO 10,100 TO 100,100
FOR i=10 TO 90 STEP 10
DRAW 10,i TO i+10,100
NEXT i
```

4.
```
FOR m=1 TO 100 STEP 2
COLOR 1
DRAW 10+m,10 TO 20+m,10
DRAW TO 10+m,20 TO 10+m,10
COLOR 0
DRAW 10+m,10 TO 20+m,10
DRAW TO 10+m,20 TO 10+m,10
NEXT m
```
(I'm sure yours was more elegant!!)

PAGE 2-3

1. a) 7/4 b) -12/19 c) 11 d) 0 e) undefined
2. (10,10),(15,12),(20,14)...many others
3. (-1,0),(2,-5),(5,-10).....
4. They would have the same slopes.

PAGE 2-5

1. a) y-6=-3(x-3) or y+3x=15 b) y-5=2/3(x+2) or 3y-2x=19
   c) y+4=-2/5(x+2) or 5y+2x=-24
2. a) y=-3x+4 b) y=5 c) x=-3
3. a) -3 b) 2/3 c) 0 d) 3 e) undefined f) -3/2
4. a) y-4=-5/8(x-3) or 8y+5x=47 b) y-6=1/4(x-2) or 4y-x=22
   c) y+10=-8/7(x-3) or 7y+8x=-46
5. y-4=-3/2(x-3) or 2y+3x=17 6. y-5=-5/3(x+4) or 3y+5x=-5

PAGE 2-10

1. a) <5,-4> b) <-9,12> c) <0,-3> d) <-6,14> e) <-3,-2>
2. <-10,-6> 3. (7,-10) 4. (0,-8)
5. 2y-x=5 and x-y=2--solve simultaneously and get x=9,y=7
   The points are (9,7) and (14,9).

6. Let $\bar{u}=\langle u1,u2\rangle$. $\bar{u}+\bar{u}=\langle u_1,u_2\rangle+\langle u_1,u_2\rangle=\langle u_1+u_2,u_1+u_2\rangle=\langle 2u_1,2u_2\rangle$

$$=2\langle u_1,u_2\rangle=2\bar{u}$$

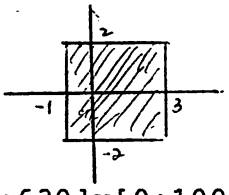
7. If t is positive, $t\bar{u}$ will be stretched(or shrunk) by a factor of t and $t\bar{u}$ will still be in the same direction. If t is

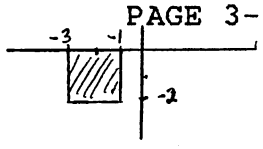
negative, $t\bar{u}$ will be in the opposite direction and stretched (or shrunk) by a factor of t .
8. Any point 9. $\sqrt{a^2+b^2}$

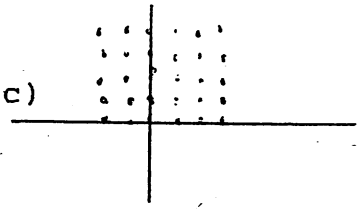
PAGE 2-12

1. x=2+5t, y=-3+7t 2. x=-3-47t, y=-7-t
3. Horiz. line from (a,b) to (c,b) is x=a+(c-a)t, y=b
   Vert. line from (a,b) to (a,c) is x=a, y=b+(c-b)t
4. If t=0 you get (x_1,y_1). If t=1 you get (x_2,x_2)
5. (7,4) and (15,1)
6. If y=2, then 2=4-3t. Thus t=2/3. X=7+8(2/3)=37/3.
   The point is (37/3,2). If y=-2, the line segment does not intersect.
7. You get points on the "extended" line segment.
8. Let (a,b) be the midpoint--for this point, t=1/2.
   So $a=x_1+(x_2-x_1)(1/2)=(x_1+x_2)/2$. Similarly $b=(y_1+y_2)/2$

PAGE 3-1

1. a) 

b) 

c) 

2. [0;639]x[0;199].
3. { (x,y,z): xε A and yε B and zε C }

PAGE 3-3

1. Suppose XL and YL are max. x & y values of screen.
   xv=(x+F)XL/2F yv=(G-y)YL/2G
2. xv=same $yv=\frac{y-C}{D-C}(YL-YS)+YS$

PAGE 4-3

1. SIN(X), COS(X), SGN(X), EXP(X), etc.
2. CHR$(X) 3. LEN(A$) 4. PLOT X,Y
5. A subroutine often accepts certain numbers or strings(input) and performs a certain action(output).
6. a) 3/2 b) -1 c) 9 7. a) 12 b) 12 c) 4.44
8. a) (2,2) b) (4,4) c) (-3,-3)
9. a) (6,1) b) (-6,7) c) (2p,4-p)
10. a) (4,2) b) (-11,-3) c) (0.1,0.35)
11 a) $\sqrt{3} \approx 1.732$ b) Undefined c) 4 d) $\sqrt{6.7} \approx 2.588$
12. a) 0 b) ≈ 0.9975 13. f(t)=(3-7t,4+t), 0≤t≤1
14. f(x)=2x+7 15. $f(x,y)=\sqrt{x^2+y^2}$
16. $f((x_1,y_2),(x_2,y_2))=\langle x_2-x_1, y_2-y_1 \rangle$

PAGE 5-2

1. a) $(x-5)^2+(y-5)^2=100$ b) $(x+3)^2+(y-6)^2=4$
   c) $(x+4)^2+(y+6)^2=1/4$ d) $x^2+y^2=1$ e) $x^2+y^2=25$
2. a) (0,2);r=6 b) (4,-3);$r=\sqrt{17}$ c) (0,0);$r=\sqrt{8}$ d) (-3,4); r=8
3. $(x-5)^2+(y-7)^2=8$ 4. y=±3 ; $x=\pm\sqrt{1.75}$ 5. $y=-3\pm\sqrt{10}$

PAGE 5-8

1. a) 3/5 b) 3/5 c) 4/5 d) 1

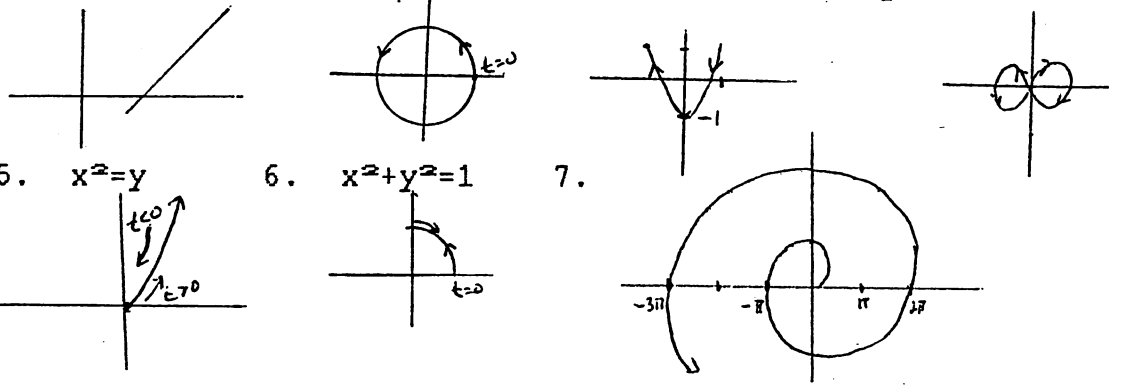
2. a) ≈0.8660 b) ≈0.9397 c) ≈0.6478 d) ≈-0.9528 e) ≈-0.8660
3. 11.47 feet.

PAGE 5-9

1. a) ≈2.356 b) ≈1.221 c) ≈1.009 d) ≈2.124 e) ≈-1.356
2. ≈11.05 3. a) $2\sin(x)\cos(x)$ b) $\cos^2(x)-\sin^2(x)$
c) $\frac{2\sin(x)\cos(x)}{\cos^2(x)-\sin^2(x)}$ (there are many equiv. expressions)
4. $\cos^2(x)-\sin^2(x) = \cos^2(x)-(1-\cos^2(x)) = 2\cos^2(x)-1$
5. $\cos A=\sqrt{3}/2$; $-\sqrt{3}/2$
6. $\sin(A+B)= 1/3 +\sqrt{15}/6$; $\cos(A+B)=\sqrt{3}/3 -\sqrt{5}/6$

PAGE 6-5

1. $2y-3x=-11$ 2. $x^2+y^2=1$ 3. $y=2x^2-1$ 4. $y^2=4x^2(1-x^2)$



5. $x^2=y$ 6. $x^2+y^2=1$ 7.

PAGE 6-9

1. a) $x^2+y^2=2y$ b) $x^2+y^2=2y+4x$ c) $y=4x$ d) $x^2-y^2=1$
e) $x=2$ f) $y^2=1+2x$
3. a) (2.6,1.5) b) (-2.2,-3.35) c) (1.6,1.2) d) (-2.85,1.45)
4. a) (8.1,60.3°), (8.1,420.3°), (-8.1,240.3°)
b) (8.9,116.6°), (8.9,-243.4°), (-8.9,296.6°)
c) (8.1,240.3°), (8.1,-119.7°), (-8.1,60.3°)

In each case, the initial angle was found from $\tan\theta = y/x$ by using the ATN(arctan or $\tan^{-1}$) function.

PAGE 8-6

1. $A+B=\begin{bmatrix}5 & 3\\ 4 & 9\end{bmatrix}$; $AB=\begin{bmatrix}12 & 13\\ 14 & 21\end{bmatrix}$; $BC=\begin{bmatrix}8 & 11 & 5\\ 26 & 34 & 32\end{bmatrix}$ $5B=\begin{bmatrix}10 & 5\\ 15 & 25\end{bmatrix}$; $BA=\begin{bmatrix}7 & 8\\ 14 & 26\end{bmatrix}$

b) No; Yes(BA=AB in part(a))
c) C is a 2x3 while B is 2x3. The number of columns in A is not the same as the number of rows in B.

2. $AB=\begin{bmatrix}16 & 3 & 8\\ 20 & 15 & 12\\ -17 & -5 & -5\end{bmatrix}$ $BA=\begin{bmatrix}9 & 2 & 0\\ 5 & 6 & 2\\ 24 & -6 & 11\end{bmatrix}$ $A^2=\begin{bmatrix}-13 & -32 & -8\\ 43 & 20 & 16\\ 10 & 10 & 11\end{bmatrix}$

3. $-A=\begin{bmatrix}-3 & 4 & -2\\ -5 & -6 & -2\\ 1 & -2 & 3\end{bmatrix}$; Define A-B = A+(-B)

4. a) AI=IA=A b) $AK=\begin{bmatrix}3 & 2 & 7\\ 2 & 1 & 4\\ 2 & 2 & 7\end{bmatrix}$ $KA=\begin{bmatrix}3 & 7 & 2\\ 2 & 7 & 2\\ 2 & 4 & 1\end{bmatrix}$

K interchanges 2nd and 3rd columns or 2nd and 3rd rows depending on the side you multiply.

c) $AN=\begin{bmatrix}-3 & 7 & 2\\ -2 & 4 & 1\\ -2 & 7 & 2\end{bmatrix}$ $NA=\begin{bmatrix}-3 & -7 & -2\\ 2 & 4 & 1\\ 2 & 7 & 2\end{bmatrix}$

N changes the signs of the 1st column or 1st row depending on the side you multiply.

PAGE 9-7

1. $\begin{vmatrix} 2 & 0 \\ 0 & 5 \end{vmatrix}$ 2. $\begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}$ 3. $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $\begin{bmatrix} .707 & -.707 \\ .707 & .707 \end{bmatrix}$

4. $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 10 & -25 & 1 \end{bmatrix}$ 5. $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ -20 & 0 & 1 \end{bmatrix}$; $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 20 & 0 & 1 \end{bmatrix}$

PAGE 11-5

1. a) $d=\sqrt{x^2+y^2+z^2}$. b) $\sqrt{120}=2\sqrt{30}$ c) $\sqrt{26}$
3. a) (4.33,2.5,10) b) (4.95,-4.95,-2) c) (4.29,2.47,4.95) d) (-7.5,4.33,5)
4. a) plane 5 units above the xy plane b) cone c) plane 4 units above xy plane d) vertical plane at 45° angle

PAGE 13-3

1. $\bar{u}\cdot\bar{v}=4$; $|\bar{u}|=\sqrt{74}$; $|\bar{v}|=\sqrt{57}$ 2. $|\bar{v}|=\sqrt{84}$
3. $u\cdot u=45$; $(1/\sqrt{45})\langle 5,4,2\rangle$ 4. $\cos\theta=\dfrac{-49}{\sqrt{45}\ \sqrt{62}}$
5. 90 degrees 6. There are many--<1,1,1> will work.($\bar{u}\cdot\bar{v}=0$)

PAGE 13-4

1. z=0 2. -(x-2)+2(y+4)+7(z-3)=0 3. 2(x-3)-5(y-5)+3(z-7)=0
4. -7(x-2)-(y-1)+10(z-3)=0 (or -7x-y+10z=15)

PAGE 13-8

1. a) 10 b) 15 c) 14 d) -1
2. 1. Interchanging the rows give the opposite value. This is implied by property (a) of cross products since if we calculate $\bar{u}$ x $\bar{v}$ and $\bar{v}$ x $\bar{u}$, the vector is in the opposite direction.
3. a) <13,1,-5> b) <-5,-7,-1> c) <-7,12,25>
4. $\bar{k}$; $\bar{i}$; $-\bar{j}$

appendix B

Graphing program that was outlined in chapter 4.
I'm sure you realize that your own efforts may not look anything like my program, and yet you may be perfectly correct. The main purpose of showing you MY program, is to give you some ideas in case you're stuck!

```
' *** General graphing program
' ***       Graphing
' ***       GFA BASIC VERSION 3
' *** variables used
'  xs,xl,ys,yl,a,b,c,d  viewport and window
' v1,v2,w1,w2  to speed calculations
' f$    function to be graphed
' answer$   response to input statement
' x,y,xv,yv   point in window and cooresp. point in viewport
' ************************************************************
LET f$="Cos(x)"           ! Edit for a different function   *
DEFFN f(x)=COS(x)         ! Edit for a different function   *
' ************************************************************
DEFTEXT 2,16   ! Red, outlined text--"Bells and whistles"
TEXT 320-LEN(f$)*4,8,UPPER$(f$)  ! center title on screen
'  default viewport and window
DATA 0,639,0,159,-12,12,-2,2
READ xs,xl,ys,yl,a,b,c,d
'
LET top=21  !top of text window
REPEAT
  REPEAT
    eraseline(top)
    PRINT "The current viewport is [";xs;",";xl;"]x[";ys;",";yl;"]"
    eraseline(top+1)
    INPUT "Do you wish to change?(y/n) ",answer$
    IF UPPER$(answer$)="Y" THEN
      eraseline(top+1)
      INPUT "enter new viewport [Xs,Xl]x[Ys,Yl] ",xs,xl,ys,yl
      eraseline(top+1)
    ENDIF
  UNTIL UPPER$(answer$)="N"
  REPEAT
    eraseline(top+1)
    PRINT "The current window is [";a;",";b;"]x[";c;",";d;"]"
    INPUT "Do you wish to change?(y,n) ";answer$
    IF UPPER$(answer$)="Y" THEN
      eraseline(top+2)
      INPUT "Enter new window [A,B]x[C,D] ",a,b,c,d
      eraseline(top+2)
    ENDIF
  UNTIL UPPER$(answer$)="N"
  ' following will speed up windowing function caluculations
  LET w1=b-a
  LET w2=d-c
  LET v1=xl-xs
  LET v2=ys-yl
  ' Draw outline around viewport
  BOX xs,ys,xl,yl
  '
```

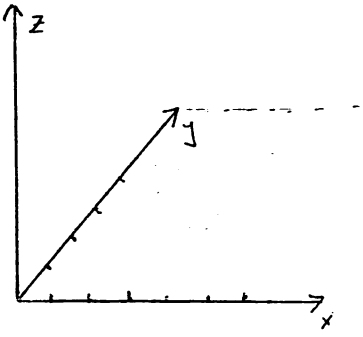
```
  ' Draw in axis if they should be showing
   ' Have adapted windowing function directly to give
   ' the viewport coord. of axis
   IF a<0 AND b>0 THEN
     DRAW -a/w1*v1+xs,ys TO -a/w1*v1+xs,yl
   ENDIF
   IF c<0 AND d>0 THEN
     DRAW xs,-c/w2*v2+yl TO xl,-c/w2*v2+yl
   ENDIF
   ' now draw curve
   FOR x=a TO b STEP w1/v1
     LET y=FN f(x)
     IF y<=d AND y>=c THEN  !point in window
       LET xv=(x-a)/w1*v1+xs
       LET yv=(y-c)/w2*v2+yl
       PLOT xv,yv
     ENDIF
   NEXT x
   eraseline(top+2)
   INPUT "Repeat for another viewport/window?(y/n) ",answer$
   eraseline(top+2)
 UNTIL UPPER$(answer$)="N"
 ' ********************************************************
 PROCEDURE eraseline(line)
   PRINT AT(1,line);SPACE$(79)
   PRINT AT(1,line);  ! reposition cursor at beginning of line
 RETURN
```

Also included on the disk is a graphing program that uses the mouse to outline the viewport. It is called MOUSGRAF.

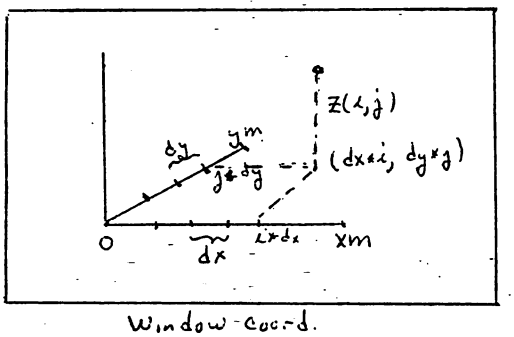
appendix C

As promised in chapter 10, here is a general program to graph a function of two variables as a two-dimensional surface. The neat thing about this program is that it turned out to be a pretty short program. It doesn't have a lot of bells and whistles--no clipping, no true perspective, only graphs over the x-y plane in the first quadrant. Yet it gives some nice results!

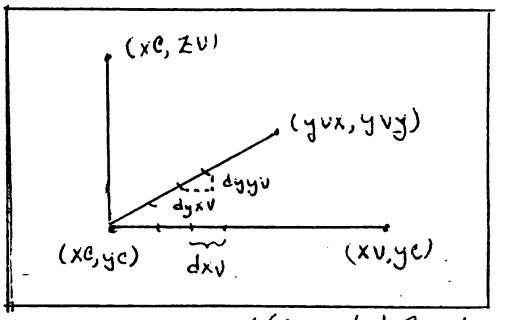
The axis for this program are a bit unorthodox. I have the y-axis going back into the screen, as shown to the right. Use your right-hand rule to satisfy yourself that it is a right-handed coordinate system. The domain of a function of two variables can be thought of as the x-y plane. Of course we can't compute and plot points for all values in the x-y plane, so we lay out a checkerboard pattern and pick points at each intersection. The variables xdiv and ydiv specify how many subdivisions of the x and y axis we want.



Below is a picture of the "window" and the viewport and the meaning of the various variables used in the program.



Window-coord.



Viewport (Computer) Coord.

Basically, we compute values of the points at the intersection of the checkerboard in the x-y plane, and save the values in a matrix Z. The points are then converted by the windowing function and plotted. The points are adjusted to the right and up to give the effect of depth and perspective. The fact that the y-axis looks like it is to the right and up, rather than being straight back into the screen is the basis of the adjustment. The variables dyxv and dyyv provide this adjustment.

```
' Deffn F(X,Y)=Sin(0.3*X)*Sin(0.3*Y)
' Deffn F(X,Y)=(X/4)^3-X/5*(Y/4)^2
DEFFN f(x,y)=COS(x*y)+1
INPUT "Enter no. of div. in x-direction,y-direction ",xdiv,ydiv
CLS
DIM z(xdiv,ydiv)
DATA Xm,4,Ym,3,Zm,4,Xc,10,Yc,180,Xv,300,Yvx,300,Yvy,100,Zv,50
READ a$,xm,a$,ym,a$,zm,a$,xc,a$,yc,a$,xv,a$,yvx,a$,yvy,a$,zv
LET dx=xm/xdiv
LET dy=ym/ydiv
LET dxv=(xv-xc)/xdiv
LET dyxv=(yvx-xc)/ydiv
LET dyyv=(yvy-yc)/ydiv
COLOR 3
DRAW xc,yc TO xv,yc
DRAW xc,yc TO yvx,yvy
DRAW xc,yc TO xc,zv
'
FOR i=0 TO xdiv
  FOR j=0 TO ydiv
    z(i,j)=FN f(dx*i,dy*j)
  NEXT j
NEXT i
'
COLOR 1
FOR i=0 TO xdiv
  PLOT i*dxv,FN window(z(i,0))
  FOR j=1 TO ydiv
    DRAW  TO i*dxv+j*dyxv,FN window(z(i,j))+j*dyyv
  NEXT j
NEXT i
FOR j=0 TO ydiv
  PLOT j*dyxv,FN window(z(0,j))+j*dyyv
  FOR i=1 TO xdiv
    DRAW  TO i*dxv+j*dyxv,FN window(z(i,j))+j*dyyv
  NEXT i
NEXT j
'
DEFFN window(zw)=zw/zm*(zv-yc)+yc
PRINT AT(1,25);
INPUT "",a$
```

appendix D

You may wish to print out the listing of RPNSUB2.LST and examine the routines. If added to your program, you may interactively enter in functions as an input string. If you remove the remark symbols from the first few lines of the "program" you can run a small demo program. There is also a graphing program, called INTERGRF that utilizes the subroutines.

RPNSUB2.LST is a subroutine that converts a string that looks like a function into an RPN stack. The stack is then evaluated when values are desired. The program works pretty fast, not as fast as GFA's built-in machine language evaluation of functions, but we can't expect that. One of my favorite gripes is that none of the languages has a built-in interactive function input. I can't believe it'd be that difficult--the gut's of evaluating strings has to be already there in the language! Feel free to incorporate the subroutine in your own programs or improve them--I don't know how thoroughly the error checking will catch weird "mistake" entries.